Training Generative Adversarial Networks in One Stage

Chengchao Shen[1], Youtan Yin[1], Xinchao Wang[2,5], Xubin Li[3], Jie Song[1,4,*], Mingli Song[1]
[1]Zhejiang University, [2]National University of Singapore, [3]Alibaba Group,
[4]Zhejiang Lab, [5]Stevens Institute of Technology
{chengchaoshen,youtanyin,sjie,brooksong}@zju.edu.cn,
xinchao@nus.edu.sg,lxb204722@alibaba-inc.com

Abstract

Generative Adversarial Networks (GANs) have demonstrated unprecedented success in various image generation tasks. The encouraging results, however, come at the price of a cumbersome training process, during which the generator and discriminator are alternately updated in two stages. In this paper, we investigate a general training scheme that enables training GANs efficiently in only one stage. Based on the adversarial losses of the generator and discriminator, we categorize GANs into two classes, Symmetric GANs and Asymmetric GANs, and introduce a novel gradient decomposition method to unify the two, allowing us to train both classes in one stage and hence alleviate the training effort. We also computationally analyze the efficiency of the proposed method, and empirically demonstrate that, the proposed method yields a solid $1.5\times$ acceleration across various datasets and network architectures. Furthermore, we show that the proposed method is readily applicable to other adversarial-training scenarios, such as data-free knowledge distillation. The code is available at https://github.com/zju-vipa/OSGAN.

1. Introduction

Generative Adversarial Networks (GANs), since their introduction in [17], have produced unprecedentedly impressive results on various image generation tasks. Thanks to the adversarial nature of the two key components, generator and discriminator, the synthesized images delivered by GANs turn out visually appealing and in many cases indistinguishable from real ones. Recently, many variants of GANs have introduced and focused on different aspects of the design, including image quality [30, 6, 12], training stability [43, 74, 46] and diversity [8, 68, 40]. Apart from generating images as the end goal, GANs have also been applied to other tasks, such as data-free knowledge distillation [44, 13, 60, 71] and domain adaption [14, 56].

[*]Corresponding author

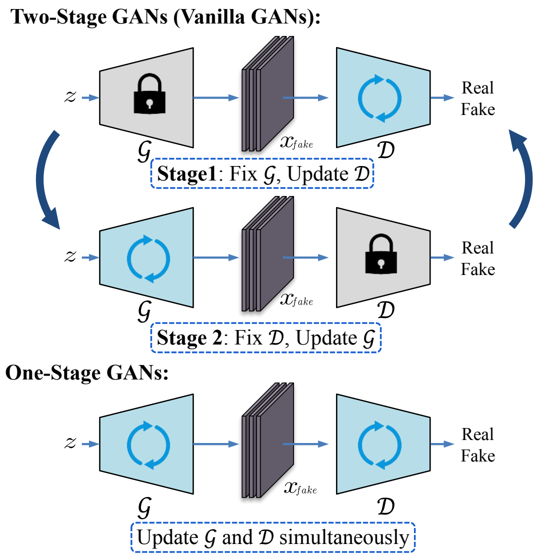
Two-Stage GANs (Vanilla GANs):
z
x_{fake}
Real
Fake
$\mathcal{G}$
$\mathcal{D}$
Stage1: Fix $\mathcal{G}$, Update $\mathcal{D}$
Stage 2: Fix $\mathcal{D}$, Update $\mathcal{G}$
One-Stage GANs:
Update $\mathcal{G}$ and $\mathcal{D}$ simultaneously

Figure 1: Comparison of the conventional Two-Stage GAN training scheme (TSGANs) and the proposed One-Stage strategy (OSGANs). The former one relies on alternately freezing the generator and the discriminant, while the latter trains both simultaneously.

The promising results delivered by GANs, however, come at the price of a burdensome training process. As shown in the upper row of Fig. 1, existing GANs rely on a time-consuming two-stage training process, which we term as *Two-Stage GANs* (TSGANs). In the first stage, fake images synthesized by the generator, together with the real ones, are fed into the discriminator for training; during this process, the discriminator is updated but the generator is fixed. In the second stage, the discriminator delivers the gradients derived from the loss function to the generator, during which the generator is updated but the discriminator is fixed. Within each adversarial round, therefore, both the generator and the discriminator carry out the feed-forward step for two times, while the discriminator implements a backward-propagation step for another two times, which, as will be analyzed in our method section, involves many repetitive computations.

Endeavors have been made towards alleviating the cumbersome training process of GANs. The work of [14], for example, adopts an efficient adversarial training strategy for unsupervised domain adaption, where learning the feature extractor and classifier requires only one round of forward inference and back-propagation. The approach of [48] also exploits a single-step optimization to update the parameters of the generator and discriminator in one turn, and showcase its power in generating visually realistic images.

In spite of their enhanced efficiency, the approaches of [14, 48] limit themselves applicable to only a subset of GANs, for which their loss functions take a particular form. Specifically, within such GANs, the adversarial loss terms in both the generator and discriminator are identical; hence, we term such models as *Symmetric GANs*. Nevertheless, many other popular GANs adopt loss functions that hold different adversarial terms for the generator and discriminator, and we term these models as *Asymmetric GANs*. The speed-up optimization techniques employed by [14, 48], unfortunately, are no longer competent to handle such asymmetric models.

We propose in this paper a novel one-stage training scheme, termed as *One-Stage GANs* (OSGANs), that generalizes to both Symmetric and Asymmetric GANs. Our key idea is to integrate the optimization for generator and discriminator during forward inference, and decompose their gradients during back-propagation to respectively update them in one stage. For the Symmetric case, since the discriminant loss hold a term that is identical to the generator loss, we only need to compute the gradient of this term once and adopt it for both losses. In this way, the updates of the generator and discriminator may safely take place in one forward and backward step.

Training Asymmetric GANs is more tricky since we can no longer copy the gradients derived from the discriminator to the generator. To this end, we carefully look into the composition of the discriminator's gradients. We discover that, the gradients derived from the different adversarial terms, in reality, preserve their proportions within the total gradients from the last layer all the way back to the first layer of discriminator. This interesting property of gradients, in turn, provides us with a feasible solution to decompose the gradients of the different adversarial terms and then to update the discriminator and generator, enabling the one-stage training of Asymmetric GANs. Finally, we unify the two classes of GANs, and show that Symmetric GANs, in fact, can be treated as a degenerate case of Asymmetric GANs.

Our contribution is therefore a general one-stage training scheme, readily applicable to various GAN variants including both Symmetric and Asymmetric GANs. Computational analysis backed up with experimental results on several datasets and network architectures demonstrate that, the proposed OSGANs achieve a solid $1.5\times$ speedup over the vanilla adversarial training strategy.

2. Related Work

We briefly review here two lines of work related to ours, including GANs and those one-stage or two-stage frameworks for other vision tasks.

2.1. Generative Adversarial Networks

The pioneering work of GAN [17] introduces an adversarial strategy, where generator is trained to synthesize fake images to confuse discriminator and discriminator tries to distinguish synthesized images from real one. When the generator and discriminator converge to a competitive equilibrium, the generator can finally synthesize realistic images from latent variables.

The mainstream GAN research has been focused on improving image quality [30, 6, 12], training stability [43, 74, 46] and diversity [8, 68, 40]. Recently, GANs have demonstrated their promising results in various image-generation tasks, such as super resolution [34, 65, 3, 62, 64], image editing [76, 7, 75, 18, 20], image inpainting [49, 24, 51], and image translation [26, 77, 63].

2.2. One-Stage and Two-Stage Framework

Recent object detection methods can be categorized into one-stage and two-stage frameworks [36]. For two-stage framework [16, 15, 54, 11], a category-agnostic region proposal module is implemented to find the possible object locations in the image, and then a category-specific classifier assigns class label for each location, where multiple feed forwards are implemented. One-stage detection methods [58, 38, 53, 33], on the other hand, unify the object bounding box and class prediction in one feed forward, which significantly reduce computation cost.

Similar to object detection, instance segmentation can also be grouped into one-stage and two-stage pipeline. For two-stage framework [50, 19, 37, 9, 10], the model first performs object detection to obtain bounding box for each object instance, and then implements binary segmentation inside each bounding box to obtain final results. With the similar motivation as detection, one-stage instance segmentation [66, 5] unifies instance mask prediction and category prediction, which effectively improves inference efficiency. Other tasks such as classification, however, have been mainly relied on one-stage schemes [70, 73, 69, 27, 28].

The above one-stage methods focus on the improvement of inference efficiency for their tasks, yet our proposed method pays more attention to the improvement of training efficiency for GANs.

3. Method

In this section, we describe the proposed method for one-stage GANs in detail. We first describe the two classes of GANs, Symmetric GANs and Asymmetric GANs, and then discuss the one-stage solutions for the above cases, respectively. Finally, we analyze the speedup factor of the proposed OSGANs.

3.1. One-Stage GANs

3.1.1 Background

The vanilla GAN proposed by [17] introduces a minmax game between discriminator $\mathcal{D}$ and generator $\mathcal{G}$ to guide the generator to synthesize realistic images. The objective is expressed as:

$$\min_{\mathcal{G}} \max_{\mathcal{D}} \mathop{\mathbb{E}}_{x\sim p_d} [\log \mathcal{D}(x)] + \mathop{\mathbb{E}}_{z\sim p_z} [\log(1-\mathcal{D}(\mathcal{G}(z)))] \, , \quad (1)$$

where x denotes real sample obtained from data distribution p_d and z denotes latent variable sampled from Guassian distribution p_z. For convenience, we rewrite the above objective as separated losses for $\mathcal{D}$ and $\mathcal{G}$ as follows:

$$\begin{aligned} \mathcal{L}_{\mathcal{D}} &= -\log \mathcal{D}(x) - \log\left(1-\mathcal{D}(\mathcal{G}(z))\right), \\ \mathcal{L}_{\mathcal{G}} &= \log\left(1-\mathcal{D}(\mathcal{G}(z))\right), \end{aligned} \quad (2)$$

where $\mathcal{L}_{\mathcal{D}}$ and $\mathcal{L}_{\mathcal{G}}$ have the same adversarial term about $\mathcal{G}(z)$: $\log\left(1-\mathcal{D}(\mathcal{G}(z))\right)$. Hence, we designate such GANs as ***Symmetric GANs***.

To alleviate the gradient vanishing problem of generator, a non-saturating loss function [1] is proposed as follows:

$$\begin{aligned} \mathcal{L}_{\mathcal{D}} &= -\log \mathcal{D}(x) - \log\left(1-\mathcal{D}(\mathcal{G}(z))\right), \\ \mathcal{L}_{\mathcal{G}} &= -\log\left(\mathcal{D}(\mathcal{G}(z))\right), \end{aligned} \quad (3)$$

where their adversarial terms about $\mathcal{G}(z)$ are different. We term GANs with such losses as ***Asymmetric GANs***. The above adversarial learning scheme is just one instance. Many other adversarial formulas can be found in the supplementary material, such as WGAN [43] and LSGAN [42].

In general, the adversarial terms in GANs' objective are the ones about fake sample $\mathcal{G}(z)$, such as $-\log\left(1-\mathcal{D}(\mathcal{G}(z))\right)$ vs $\log\left(1-\mathcal{D}(\mathcal{G}(z))\right)$ in Eq. 2 and $-\log\left(1-\mathcal{D}(\mathcal{G}(z))\right)$ vs $-\log\left(\mathcal{D}(\mathcal{G}(z))\right)$ in Eq. 3. Therefore, for clarity of analysis, we split general $\mathcal{L}_{\mathcal{D}}$ into two parts: the term about real sample $\mathcal{L}^r_{\mathcal{D}}(x)$ and the term about fake sample $\mathcal{L}^f_{\mathcal{D}}(\hat{x})$, where $\hat{x} = \mathcal{G}(z)$. [*] More discussions can be found in the supplementary material. Finally, the objective of general GANs can be written as follows:

$$\begin{aligned} \mathcal{L}_{\mathcal{D}}(x,\hat{x}) &= \mathcal{L}^r_{\mathcal{D}}(x) + \mathcal{L}^f_{\mathcal{D}}(\hat{x}), \\ \mathcal{L}_{\mathcal{G}}(\hat{x}) &= \mathcal{L}_{\mathcal{G}}(\mathcal{G}(z)). \end{aligned} \quad (4)$$

[*]There are other forms of GANs, whose $\mathcal{L}_{\mathcal{D}}$ can not be explicitly split in this way. Yet still, our approach is applicable.

For brevity, we omit input x and $\hat{x}$, such as $\mathcal{L}_{\mathcal{D}}$, $\mathcal{L}^r_{\mathcal{D}}$, $\mathcal{L}^f_{\mathcal{D}}$ and $\mathcal{L}_{\mathcal{G}}$. Based on Eq. 4, we can formally distinguish Symmetric GANs and Asymmetric GANs: for Symmetric GANs, we have $\mathcal{L}_{\mathcal{G}} = -\mathcal{L}^f_{\mathcal{D}}$, while for Asymmetric GANs, we have $\mathcal{L}_{\mathcal{G}} \neq -\mathcal{L}^f_{\mathcal{D}}$.

3.1.2 Symmetric OSGANs

For Symmetric GANs (e.g. Eq.2), $\mathcal{L}_{\mathcal{G}}$ and $\mathcal{L}_{\mathcal{D}}$ contain the same loss term about fake sample $\hat{x}$: $\mathcal{L}^f_{\mathcal{D}}$. Their gradients w.r.t. $\hat{x}$ can be denoted as $\nabla_{\hat{x}}\mathcal{L}_{\mathcal{D}} = \nabla_{\hat{x}}\mathcal{L}^f_{\mathcal{D}}$ and $\nabla_{\hat{x}}\mathcal{L}_{\mathcal{G}} = -\nabla_{\hat{x}}\mathcal{L}^f_{\mathcal{D}}$, respectively. We can obtain $\nabla_{\hat{x}}\mathcal{L}_{\mathcal{G}}$ from $\nabla_{\hat{x}}\mathcal{L}_{\mathcal{D}}$ by just multiplying $\nabla_{\hat{x}}\mathcal{L}_{\mathcal{D}}$ with -1 and then compute the gradients w.r.t. parameters of $\mathcal{G}$ from $\nabla_{\hat{x}}\mathcal{L}_{\mathcal{G}}$. In a word, we can update the parameters of $\mathcal{G}$ with $\nabla_{\hat{x}}\mathcal{L}_{\mathcal{D}}$, which is computed during the training of $\mathcal{D}$. This method simplifies the training of Symmetric GANs from two stages to one stage.

3.1.3 Asymmetric OSGANs

For Asymmetric GANs (e.g. Eq.3), due to $\mathcal{L}_{\mathcal{G}} \neq -\mathcal{L}^f_{\mathcal{D}}$, the gradients $\nabla_{\hat{x}}\mathcal{L}_{\mathcal{G}}$ can not be directly obtained from $\nabla_{\hat{x}}\mathcal{L}_{\mathcal{D}}$ as Symmetric GANs.

An intuitive idea to solve this problem is to integrate $\mathcal{L}_{\mathcal{G}}$ and $\mathcal{L}_{\mathcal{D}}$ into one, e.g. $\mathcal{L}_{\mathcal{D}} \leftarrow \mathcal{L}_{\mathcal{D}} + \mathcal{L}_{\mathcal{G}}$, so that we can obtain $\nabla_{\hat{x}}\mathcal{L}_{\mathcal{G}}$ from $\nabla_{\hat{x}}\mathcal{L}_{\mathcal{D}}$. Since the sign of $\nabla_{\hat{x}}\mathcal{L}^f_{\mathcal{D}}$ and $\nabla_{\hat{x}}\mathcal{L}_{\mathcal{G}}$ is generally reversed, we adopt

$$\mathcal{L} = \mathcal{L}_{\mathcal{D}} - \mathcal{L}_{\mathcal{G}} = \mathcal{L}^r_{\mathcal{D}} + \mathcal{L}^f_{\mathcal{D}} - \mathcal{L}_{\mathcal{G}}, \quad (5)$$

instead of $\mathcal{L} = \mathcal{L}_{\mathcal{D}} + \mathcal{L}_{\mathcal{G}}$ to avoid the gradient counteraction between $\mathcal{L}^f_{\mathcal{D}}$ and $\mathcal{L}_{\mathcal{G}}$. We denote $\mathcal{L}_f = \mathcal{L}^f_{\mathcal{D}} - \mathcal{L}_{\mathcal{G}}$ to collect the loss terms about fake sample. By doing so, however, another issue is introduced: how to recover $\nabla_{\hat{x}}\mathcal{L}_{\mathcal{G}}$ from mixed gradients $\nabla_{\hat{x}}\mathcal{L}_f$.

To tackle this problem, we investigate back-propagation about the discriminator network. We find an interesting property of the mainstream neural modules. The relation between the gradients of loss $\mathcal{L}$ w.r.t. the input $\nabla_{in}\mathcal{L}$, and the gradient of loss $\mathcal{L}$ w.r.t. its output $\nabla_{out}\mathcal{L}$ can be presented as the follows:

$$\nabla_{in}\mathcal{L} = P \cdot \mathcal{F}\left(\nabla_{out}\mathcal{L}\right) \cdot Q, \quad (6)$$

where P and Q are matrices that depend on the module or its input; $\mathcal{F}(\cdot)$ is a function that satisfies the equation $\mathcal{F}(y_1+y_2) = \mathcal{F}(y_1)+\mathcal{F}(y_2)$. The neural modules include convolutional module, fully-connected module, non-linear activation operation, and pooling. Although both non-linear activation and pooling are non-linear operations, the relation between their gradients of loss function w.r.t. input and output meets Eq. 6. More details are discussed in the supplementary material.

Based on Eq. 6, we can obtain the relation between the gradients of $\mathcal{L}_{\mathcal{G}}$ and $\mathcal{L}_{\mathcal{D}}$ for fake sample $\hat{x}_i$ in discriminator as follows:

$$\frac{\nabla_{\hat{x}_i}\mathcal{L}_{\mathcal{G}}}{\nabla_{\hat{x}_i}\mathcal{L}_{\mathcal{D}}} = \cdots = \frac{\nabla_{\hat{x}_i^l}\mathcal{L}_{\mathcal{G}}}{\nabla_{\hat{x}_i^l}\mathcal{L}_{\mathcal{D}}} = \cdots = \frac{\nabla_{\hat{x}_i^L}\mathcal{L}_{\mathcal{G}}}{\nabla_{\hat{x}_i^L}\mathcal{L}_{\mathcal{D}}} = \gamma_i, \tag{7}$$

where L is the total number of layers, γ_i is an instance-wise scalar for $\hat{x}_i$,[*] and $\hat{x}_i^l$ denotes features of l-th layer in discriminator.

In other words, γ_i remains a constant for sample $\hat{x}_i^l$ cross different layers, despite each sample holds a different γ_i. On the one hand, we only need to compute γ_i for the last layer, and computing $\nabla_{\hat{x}_i^L}\mathcal{L}_{\mathcal{G}}/\nabla_{\hat{x}_i^L}\mathcal{L}_{\mathcal{D}}$ to obtain γ_i for all layers is in practice very efficient. On the other hand, it varies with different samples and therefore needs to be re-computed for each sample. Since γ_i is the ratio between two scalars $\nabla_{\hat{x}_i^L}\mathcal{L}_{\mathcal{G}}$ and $\nabla_{\hat{x}_i^L}\mathcal{L}_{\mathcal{D}}$, the cost of this tiny computation can be omitted during training.

Combining $\nabla_{\hat{x}_i^l}\mathcal{L}_f = \nabla_{\hat{x}_i^l}\mathcal{L}_{\mathcal{D}}^f - \nabla_{\hat{x}_i^l}\mathcal{L}_{\mathcal{G}}$ and Eq. 7, we can proportionally decompose $\nabla_{\hat{x}_i^l}\mathcal{L}_{\mathcal{G}}$ and $\nabla_{\hat{x}_i^l}\mathcal{L}_{\mathcal{D}}^f$ from mixed gradients: $\nabla_{\hat{x}_i^l}\mathcal{L}_f$:

$$\begin{aligned}\nabla_{\hat{x}_i^l}\mathcal{L}_{\mathcal{D}}^f &= \frac{1}{1-\gamma_i}\nabla_{\hat{x}_i^l}\mathcal{L}_f,\\ \nabla_{\hat{x}_i^l}\mathcal{L}_{\mathcal{G}} &= \frac{\gamma_i}{1-\gamma_i}\nabla_{\hat{x}_i^l}\mathcal{L}_f.\end{aligned} \tag{8}$$

In other words, we can obtain $\nabla_{\hat{x}_i^l}\mathcal{L}_{\mathcal{G}}$ and $\nabla_{\hat{x}_i^l}\mathcal{L}_{\mathcal{D}}^f$ by scaling $\nabla_{\hat{x}_i^l}\mathcal{L}_f$. For the convenience of computation, we apply the above scaling operation to loss function $\mathcal{L}_f$, which is equivalent to Eq. 8. Hence, we can obtain the instance loss functions for discriminator and generator:

$$\begin{aligned}\mathcal{L}_{\mathcal{D}}^{ins} &= \mathcal{L}_{\mathcal{D}}^r + \frac{1}{1-\gamma_i}\left(\mathcal{L}_{\mathcal{D}}^f - \mathcal{L}_{\mathcal{G}}\right),\\ \mathcal{L}_{\mathcal{G}}^{ins} &= \frac{\gamma_i}{1-\gamma_i}\left(\mathcal{L}_{\mathcal{D}}^f - \mathcal{L}_{\mathcal{G}}\right),\end{aligned} \tag{9}$$

where both $\mathcal{L}_{\mathcal{D}}^{ins}$ and $\mathcal{L}_{\mathcal{G}}^{ins}$ contain the same loss term $(\mathcal{L}_{\mathcal{D}}^f - \mathcal{L}_{\mathcal{G}})$. In this way, Asymmetric GAN is transformed into a symmetric one. Hence, a similar one-stage training strategy adopted in Symmetric OSGANs can be applied. The difference is that the gradients from $\mathcal{L}_{\mathcal{D}}^{ins}$ needs to be scaled by γ_i instead of -1, since $\nabla_{\hat{x}_i^l}\mathcal{L}_{\mathcal{G}}^{ins} = \gamma_i \cdot \nabla_{\hat{x}_i^l}\mathcal{L}_{\mathcal{D}}^{ins}$.

A numerical detail needs to considered in Eq. 9: if $(1-\gamma_i)$ is very close to zero, a numerical instability issue occurs. Due to the competition between $\mathcal{L}_{\mathcal{D}}^f$ and $\mathcal{L}_{\mathcal{G}}$, the signs of $\nabla_{x_i^l}\mathcal{L}_{\mathcal{D}}^f$ and $\nabla_{x_i^l}\mathcal{L}_{\mathcal{G}}$ tend to be reversed. Hence, γ_i is a negative number, which ensures the numerical stability.

[*]The derivation can be found in the supplementary material. In fact, γ_i is a tensor with the same size as $\hat{x}_i^l$, in which all elements have the same value. For brevity, we regard it as a scalar.

Algorithm 1 One-Stage GAN Training Framework

```
Input: Training data $\mathcal{X} = \{x_j\}_{j=1}^N$.
Output: The parameters of generator $\mathcal{G}$.
 1: Initialize the parameters of $\mathcal{G}$ and $\mathcal{D}$;
 2: for number of training iterations do
 3:     Sample $z \sim \mathcal{N}(0,1)$ to generate fake data $\hat{x}_i$ by $\mathcal{G}$;
 4:     Sample real data $x_j$ from $\mathcal{X}$;
 5:     Feed $\hat{x}_i$ and $x_j$ into $\mathcal{D}$ to compute $\mathcal{L}_{\mathcal{G}}$ and $\mathcal{L}_{\mathcal{D}}$;
 6:     Compute $\nabla_{\hat{x}_i^L}\mathcal{L}_{\mathcal{G}}$ and $\nabla_{\hat{x}_i^L}\mathcal{L}_{\mathcal{D}}$ to obtain $\gamma_i$ by Eq. 7;
 7:     Compute $\mathcal{L}_{\mathcal{D}}^{ins}$ by Eq. 9;
 8:     Back propagate $\nabla_{\hat{x}_i^L}\mathcal{L}_{\mathcal{D}}^{ins}$ to obtain $\nabla_{\hat{x}_i}\mathcal{L}_{\mathcal{D}}^{ins}$;
 9:     Obtain $\nabla_{\hat{x}_i}\mathcal{L}_{\mathcal{G}}^{ins} = \gamma_i \cdot \nabla_{\hat{x}_i}\mathcal{L}_{\mathcal{D}}^{ins}$ by Eq. 7;
10:     Update $\mathcal{G}$ and $\mathcal{D}$ simultaneously using Adam.
11: end for
```

Let us now take a further look into the relation between solutions for Symmetric OSGANs and Asymmetric OSGANs. Based on Eq. 7, when $\mathcal{L}_{\mathcal{G}} = -\mathcal{L}_{\mathcal{D}}^f$ is met, we have $\gamma_i = -1$. Combining $\gamma_i = -1$ and Eq. 9, we can obtain $\mathcal{L}_{\mathcal{D}}^{ins} = \mathcal{L}_{\mathcal{D}}^r + \mathcal{L}_{\mathcal{D}}^f$ and $\mathcal{L}_{\mathcal{G}}^{ins} = -\mathcal{L}_{\mathcal{D}}^f$, which has the same form as Symmetric OSGAN. In other words, our method is a general solution for both Symmetric GANs and Asymmetric GANs. The overall algorithm for our proposed approach can be summarized as Algorithm 1.

3.2. Efficiency Analysis

In this section, we computationally analyze the efficiency of OSGANs and TSGANs. To this end, we investigate three perspectives: (1) the training time for real samples and fake ones in one batch, (2) the time for forward inference and back-propagation, (3) the time for parameter gradient computation and back-propagation.

For the training time, we assume that the batch size of real samples and fake ones is equal, which is a widely adopted setting in practice. Hence, the time for real samples is equal to the time for fake ones, which works on both forward inference and back-propagation:

$$\begin{aligned}\mathcal{T}_{forw}(x_r) &= \mathcal{T}_{forw}(x_f),\\ \mathcal{T}_{back}(x_r) &= \mathcal{T}_{back}(x_f).\end{aligned} \tag{10}$$

For the forward and backward time, we focus on the cost on modules, such as convolutional and fully-connected ones, which take up the majority of time. We find the time cost on forward inference and back-propagation for these modules is approximately equal, which will be discussed in the supplementary material. For simplicity, the above time is represented as $\mathcal{T}(x)$:

$$\mathcal{T}_{forw}(x) = \mathcal{T}_{back}(x) = \mathcal{T}(x). \tag{11}$$

For linear modules, such as convolutional and fully-connected layers, due to the equal number of floating point

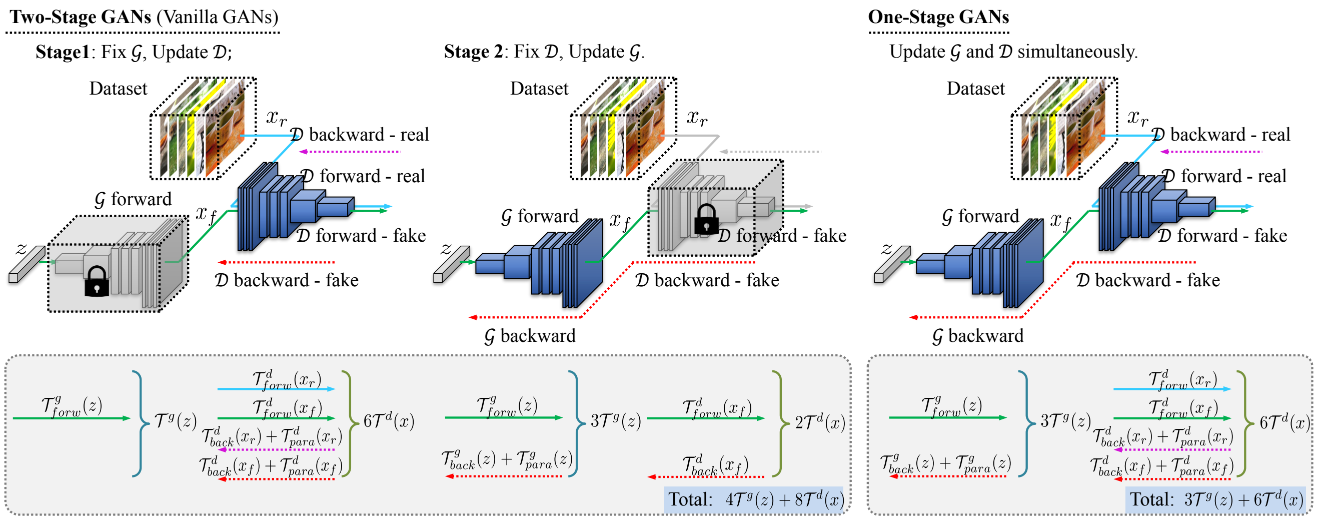
Two-Stage GANs (Vanilla GANs)
Stage1: Fix $\mathcal{G}$, Update $\mathcal{D}$;
Stage 2: Fix $\mathcal{D}$, Update $\mathcal{G}$.
One-Stage GANs
Update $\mathcal{G}$ and $\mathcal{D}$ simultaneously.
Dataset
x_r
x_f
z
$\mathcal{D}$ backward - real
$\mathcal{D}$ forward - real
$\mathcal{D}$ forward - fake
$\mathcal{D}$ backward - fake
$\mathcal{G}$ forward
$\mathcal{G}$ backward
$\mathcal{T}^g_{forw}(z)$
$\mathcal{T}^g(z)$
$\mathcal{T}^d_{forw}(x_r)$
$\mathcal{T}^d_{forw}(x_f)$
$\mathcal{T}^d_{back}(x_r)+\mathcal{T}^d_{para}(x_r)$
$\mathcal{T}^d_{back}(x_f)+\mathcal{T}^d_{para}(x_f)$
$6\mathcal{T}^d(x)$
$\mathcal{T}^g_{back}(z)+\mathcal{T}^g_{para}(z)$
$3\mathcal{T}^g(z)$
$\mathcal{T}^d_{back}(x_f)$
$2\mathcal{T}^d(x)$
Total: $4\mathcal{T}^g(z)+8\mathcal{T}^d(x)$
Total: $3\mathcal{T}^g(z)+6\mathcal{T}^d(x)$

Figure 2: Efficiency comparison between TSGANs (left) and OSGANs (right). For TSGANs, in the first stage, $\mathcal{D}$ is trained to classify the fake and real samples, where $\mathcal{G}$ forwards one time and $\mathcal{D}$ forwards and backwards one time for real and fake samples, respectively. In the second stage, $\mathcal{G}$ is trained to confuse $\mathcal{D}$, where both $\mathcal{G}$ and $\mathcal{D}$ backward and forward one time. For OSGANs, $\mathcal{G}$ and $\mathcal{D}$ are trained simultaneously, where both of them only need forward and backward one time.

operations, the time consumed on parameter gradient computation and backward is nearly identical. We write:

$$\mathcal{T}_{para}(x) = \mathcal{T}_{back}(x) = \mathcal{T}(x). \tag{12}$$

To estimate the low bound of speedup ratio for our method, we assume all modules in $\mathcal{G}$ and $\mathcal{D}$ contain learnable parameters, for which the gradients need computing. Under this assumption, the efficiency is analyzed as follow.

As shown in Fig. 2, for TSGANs, in the first stage, discriminator is trained to discriminate the fake samples synthesized by generator and the real ones. The generator takes $\mathcal{T}^g_{forw}(z)$ to generate samples. And the discriminator takes $\mathcal{T}^d_{forw}(x_r) + \mathcal{T}^d_{forw}(x_f)$ on forward inference, $\mathcal{T}^d_{back}(x_r) + \mathcal{T}^d_{back}(x_f)$ on back-propagation and $\mathcal{T}^d_{para}(x_r)+\mathcal{T}^d_{para}(x_f)$ on parameter gradients computation. Combined with Eq. 10, Eq. 11 and Eq. 12, the time for the first stage is

$$\mathcal{T}_{stage1}(z, x) = \mathcal{T}^g(z) + 6\mathcal{T}^d(x), \tag{13}$$

where both $\mathcal{T}^g(z)$ and $\mathcal{T}^d(x)$ denote the time cost by generator and discriminator on forward inference and back-propagation, respectively.

In the second stage, the generator takes $\mathcal{T}^g_{forw}(z)$, $\mathcal{T}^g_{back}(z)$ and $\mathcal{T}^g_{para}(z)$ on forward inference, back-propagation and parameter gradients computation, respectively. The discriminator takes $\mathcal{T}^d_{forw}(x_f)$ and $\mathcal{T}^d_{back}(x_f)$ on forward inference and back-propagation, respectively. Combined with Eq. 10 and Eq. 11, the time for the second stage is

$$\mathcal{T}_{stage2}(z, x) = 3\mathcal{T}^g(z) + 2\mathcal{T}^d(x). \tag{14}$$

Overall, the total time for TSGANs is therefore

$$\begin{aligned}\mathcal{T}_{two}(z, x) &= \mathcal{T}_{stage1}(z, x) + \mathcal{T}_{stage2}(z, x)\\ &= 4\mathcal{T}^g(z) + 8\mathcal{T}^d(x).\end{aligned} \tag{15}$$

In the same way, the time for OSGANs is

$$\mathcal{T}_{one}(z, x) = 3\mathcal{T}^g(z) + 6\mathcal{T}^d(x). \tag{16}$$

Combining Eq. 15 and Eq. 16, we can obtain the training speedup ratio of OSGANs against TSGANs in the worst case:

$$S = \frac{\mathcal{T}_{two}(z, x)}{\mathcal{T}_{one}(z, x)} = \frac{4\mathcal{T}^g(z) + 8\mathcal{T}^d(x)}{3\mathcal{T}^g(z) + 6\mathcal{T}^d(x)} = \frac{4}{3}. \tag{17}$$

4. Experiments

In this section, we conduct experiments to evaluate the effectiveness of our proposed training method. First, we introduce the experimental settings, including the datasets we used, evaluation metrics and the implementation details. We then show experimental results on popular generation benchmarks including quantitative analysis. Afterwards, we compare the training efficiency between one-stage training and two-stage one. Finally, we adopt our one-stage training strategy on data-free adversarial distillation.

4.1. Experimental Settings

4.1.1 Datasets

CelebFaces Attributes Dataset (CelebA) [39] is a large-scale face attributes dataset, which consists of more than 200K celebrity images. In the experiment, the images are

	CelebA		LSUN Churches		FFHQ	
Method	**FID**	**KID**	**FID**	**KID**	**FID**	**KID**
DCGAN [52] (sym, two)	23.78±0.12	0.019±0.001	46.09±0.25	0.043±0.002	40.56±0.32	0.040±0.001
DCGAN (sym, one)	22.66±0.09(↓)	0.018±0.001(↓)	38.99±0.21(↓)	0.037±0.001(↓)	29.38±0.24(↓)	0.026±0.001(↓)
CGAN [45] (sym, two)	39.85±0.27	0.035±0.000	/	/	/	/
CGAN (sym, one)	30.97±0.33(↓)	0.013±0.000(↓)	/	/	/	/
WGAN [43] (sym, two)	33.30±0.23	0.028±0.001	36.29±0.14	0.034±0.002	35.66±0.13	0.032±0.001
WGAN (sym, one)	27.23±0.18(↓)	0.021±0.001(↓)	35.58±0.30(↓)	0.033±0.002(↓)	39.77±0.14(↑)	0.036±0.001(↑)
DCGAN [1]† (asym, two)	25.34±0.20	0.022±0.001	33.35±0.40	0.030±0.001	31.50±0.15	0.031±0.001
DCGAN† (asym, one)	24.20±0.17(↓)	0.019±0.001(↓)	25.41±0.30(↓)	0.024±0.001(↓)	31.19±0.21(↓)	0.028±0.001(↓)
LSGAN [41] (asym, two)	23.00±0.17	0.019±0.001	22.63±0.16	0.021±0.001	30.29±0.23	0.029±0.001
LSGAN (asym, one)	19.31±0.18(↓)	0.015±0.001(↓)	22.39±0.37(↓)	0.019±0.001(↓)	29.11±0.12(↓)	0.026±0.001(↓)
GeoGAN [35] (asym, two)	21.92±0.13	0.018±0.001	18.85±0.41	0.016±0.001	30.55±0.21	0.031±0.001
GeoGAN (asym, one)	21.52±0.14(↓)	0.017±0.001(↓)	18.92±0.24(↑)	0.017±0.001(↑)	30.63±0.26(↑)	0.032±0.001(↑)
RelGAN [29] (asym, two)	20.91±0.15	0.018±0.001	24.95±0.19	0.023±0.002	35.81±0.18	0.033±0.001
RelGAN (asym, one)	20.79±0.18(↓)	0.015±0.001(↓)	25.09±0.25(↑)	0.024±0.001(↑)	35.72±0.16(↓)	0.031±0.001(↓)
FisherGAN [47] (asym, two)	27.88±0.29	0.024±0.001	29.25±0.32	0.026±0.001	52.65±0.16	0.050±0.001
FisherGAN (asym, one)	27.10±0.19(↓)	0.021±0.001(↓)	29.23±0.18(↓)	0.025±0.001(↓)	51.63±0.18(↓)	0.049±0.001(↓)
BGAN [23] (asym, two)	31.12±0.19	0.027±0.001	35.06±0.31	0.035±0.001	41.35±0.17	0.038±0.001
BGAN (asym, one)	25.34±0.17(↓)	0.022±0.001(↓)	34.42±0.21(↓)	0.035±0.001(↓)	42.64±0.13(↑)	0.039±0.001(↑)
SNGAN [46] (asym, two)	34.95±0.24	0.033±0.001	33.51±0.16	0.034±0.001	52.46±0.24	0.050±0.001
SNGAN (asym, one)	34.34±0.12(↓)	0.033±0.001(↓)	31.23±0.17(↓)	0.031±0.001(↓)	51.03±0.36(↓)	0.048±0.001(↓)

Table 1: Comparative results of OSGANs and TSGANs. All results are averaged over five runs and error bars correspond to the standard deviation. Specifically, "sym" and "asym" respectively denote Symmetric GANs and Asymmetric GANs, while "one" and "two" respectively denote one-stage and two-stage strategy. † adopts asymmetric adversarial loss in [1]; ↓ and ↑ respectively denote our strategy outperforms and underperforms the two-stage one.

roughly aligned according to the two eye locations and then resized and cropped into 64×64 size.

LSUN churches [72] is a dataset that contains more than 126K church outdoor images. Flickr-Faces-HQ (FFHQ) [31] is a face dataset, which consists of 70K images and covers large variation in term of age, ethnicity, image background and accessory. For training convenience, all images of the above dataset are resized and cropped into 64×64 resolution.

Both CIFAR10 and CIFAR100 [32] are composed of 60,000 colour images with 32×32 size, where 50,000 images are used as training set and the rest 10,000 images are used as test set. The CIFAR10 dataset contains 10 classes, and the CIFAR100 contains 100 classes. Random cropping and random horizontal flipping are applied to the training images to augment the dataset.

4.1.2 Implementation Details

The proposed method is implemented using PyTorch v1.5 on a Quadro P5000 16G GPU. The batch size for both real images and fake one is set to 128. In all experiments, Adam optimization algorithm is adopted for both $\mathcal{G}$ and $\mathcal{D}$. For the convenience of experiments, we adopt the network architecture used by DCGANs as the backbone. Given that Eq. 6 does not work on batch normalization [25], we replace the batch normalization in $\mathcal{D}$ with group normalization [67] so as to meet Eq. 6.

4.1.3 Evaluation Metrics

To evaluate the performance of GANs, three popular evaluation metrics are available.

Fréchet Inception Distance (FID) [21] is used to evaluate the distance between features from real images and the ones from generated images, which is computed as follows:

$$\text{FID} = \|\mu_r - \mu_g\| + \text{tr}\left(C_r + C_g - 2\left(C_r C_g\right)^{1/2}\right), \quad (18)$$

where μ_r and μ_g are the empirical means for real and generated samples, C_r and C_g are the empirical covariance for real and generated samples, respectively.

Kernel Inception Distance (KID) [4] mitigates the overfitting problem, which may occur in FID. It is computed by

$$\text{KID} = \mathop{\mathbb{E}}_{x_r, x_g}\left[k\left(x_r, x_r'\right) - 2k\left(x_r, x_g\right) + k\left(x_g, x_g'\right)\right], \quad (19)$$

where x_r and x_g are features for real and generated samples, respectively. $k\left(x, y\right) = \left(\frac{1}{d}x^{\mathrm{T}}y + 1\right)^3$.

Inception Score (IS) [57] is proposed to measure the realistic level of a generated image. As discussed in [2] and [55], applying the IS to generative models trained on datasets other than ImageNet gives misleading results. The IS should only be used as a "rough guide" to evaluate generative models. Hence, in the following comparative experiments, we will focus on FID and KID.

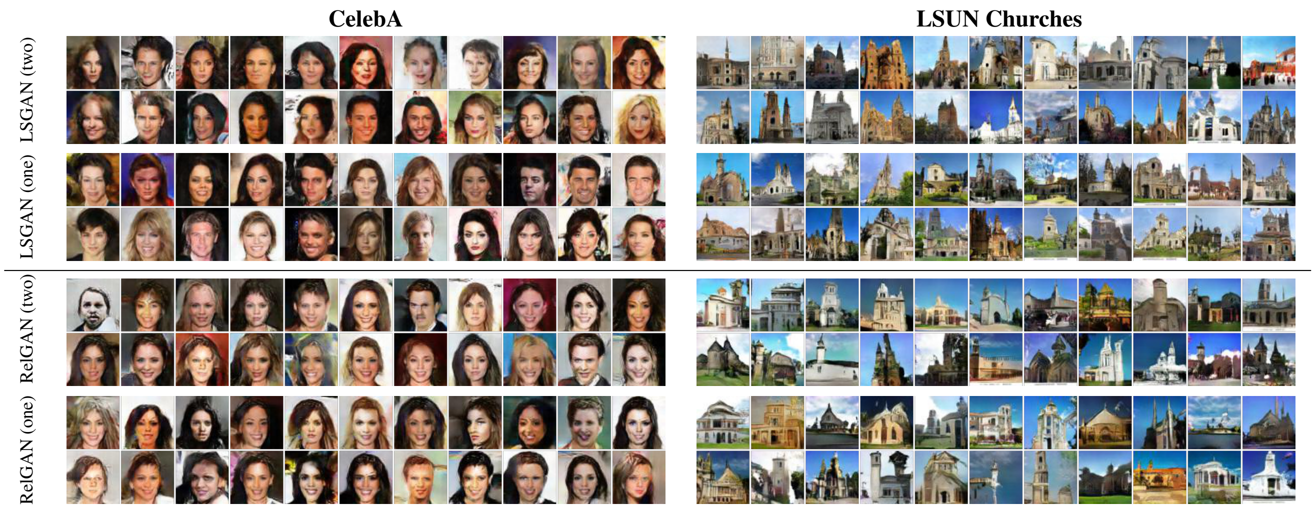

Figure 3: Results synthesized by TSGANs ("two") and OSGANs ("one") on CelebA and LSUN Churches.

4.2. One-Stage Adversarial Training

To verify the effectiveness of one-stage adversarial training, we compare the performance on several representative GANs, each of which is trained with one-stage strategy and two-stage one, respectively. The comparative results, including those obtained by symmetric GANs and asymmetric GANs, are shown in Tab. 1,

Symmetric GANs trained with one-stage training strategy, in reality, outperform their two-stage counterparts, except for WGAN on FFHQ. We analyze a possible reason as follows. In two-stage training, the gradients of generator are computed after the training of discriminator. This asynchronous gradient computation may lead to inefficient optimization, where the earlier optimization information is lost during the training of generator. The one-stage training, on the other hand, effectively bypasses the problem by synchronously updating the generator and discriminator.

Variants of Asymmetric GANs with different asymmetric adversarial objectives are evaluated. On DCGAN, LSGAN, FisherGAN and SNGAN, our one-stage strategy consistently outperforms their two-stage counterparts on all datasets. The performance of one-stage GeoGAN, RelGAN and BGAN is on par with their corresponding two-stage one. This can be in part explained by that, due to the adversarial-objective difference between generator and discriminator, the effect of optimization information loss is not so evident. Overall, our one-stage strategy training achieves performance comparable to those of the existing two-stage ones. Some visualization results are shown in Fig. 3. More results can be found in the supplementary material.

4.3. Efficiency Analysis

In this section, we empirically evaluate the efficiency of OSGANs against TSGANs. As shown in Fig. 4, both Symmetric GAN (DCGAN-sym) and Asymmetric GAN (DCGAN-asym and LSGAN-asym) are compared with the corresponding TSGANs, respectively. Due to small value of KID, we take logarithm of KID for better visualization.

For Symmetric DCGAN, one-stage training strategy tends to be more stable than two-stage one. For Symmetric GAN, the end-to-end training strategy significantly smooths the gradients for optimization, which leads to a more stable training process. One-stage Symmetric DCGAN outperforms the two-stage one on all datasets, while it achieves a speedup ratio sometimes greater than 1.5 (e.g. $458.7/267.5 \approx 1.7$). We believe that one-stage training adopts the synchronous update for the generator and discriminator, which can significantly avoid the early gradient saturation in two-stage strategy.

For asymmetric DCGAN, one-stage strategy achieves faster convergence speed and better performance than two-stage one on CelebA and LSUN Churches (speedup ratio: $286.1/180.5 \approx 1.6$). For asymmetric LSGAN, one-stage training consistently converges faster than the two-stage one on all datasets (speedup ratio: $286.4/186.0 \approx 1.5$). In the meantime, one-stage training has smaller fluctuation than the two-stage one. More results can be found in the supplementary material.

4.4. Application: Knowledge Distillation

We also investigate the effectiveness of our training strategy on adversarial knowledge distillation task [59, 61]. Both DFAD [13] and ZSKT [44] adopt an adversarial training strategy to implement knowledge distillation without real data, where teacher is ResNet34 and the student is ResNet18. Specifically, in each adversarial round, the student learns to imitate the prediction of the teacher by five iterations, while the generator tends to synthesize harder

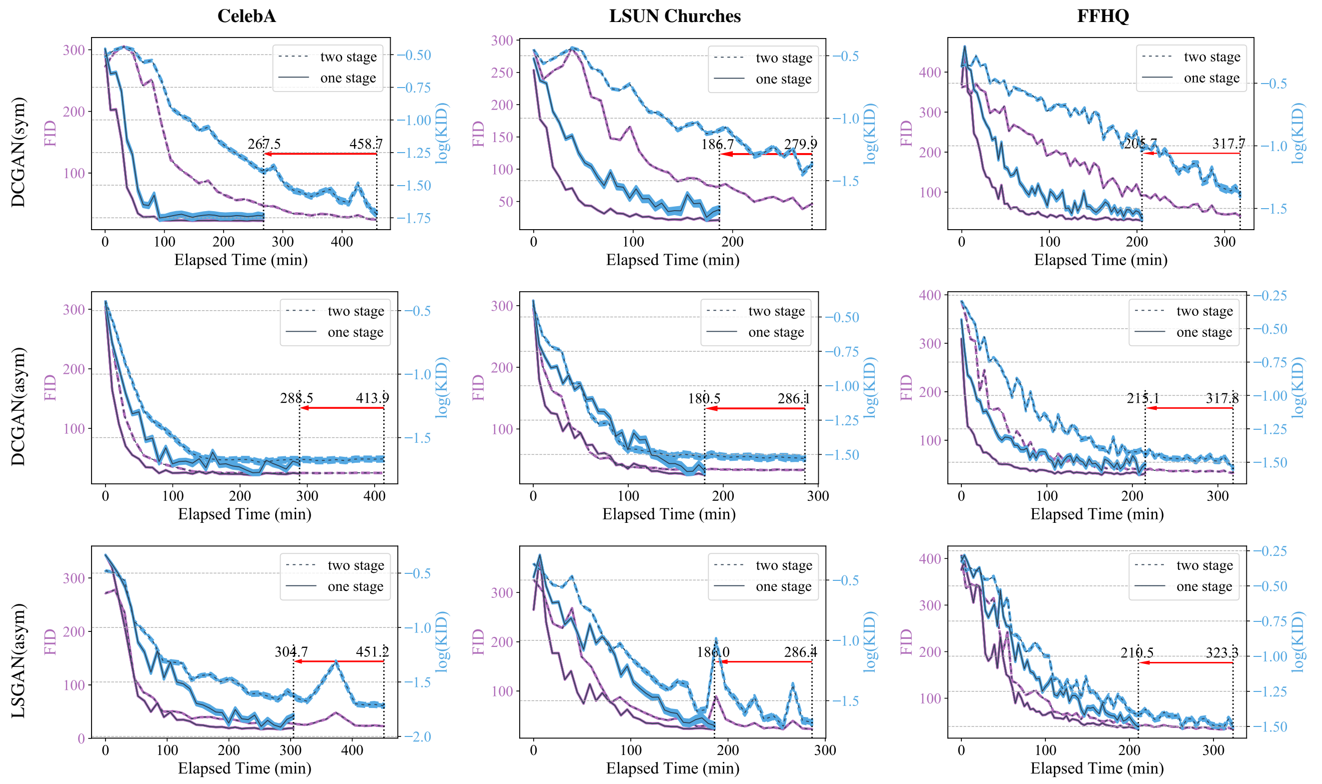

Figure 4: Comparison of training efficiency between OSGANs and TSGANs. All experiments are terminated when their performance reaches plateau.

samples with larger loss values by one iteration. In other words, it adopts a two-stage adversarial training strategy. Using our proposed method, the adversarial distillation achieves competition within one stage. There is only one update on generator and student network in each round, respectively. Experimental results are shown in Tab. 2, demonstrating that our proposed method achieves performance on par with the original two-stage one, sometimes even better. Especially on CIFAR100, our method outperforms DFAD and ZSKT by a large margin, significantly reducing the performance gap between knowledge distillation using full dataset and data-free knowledge distillation.

	CIFAR10		**CIFAR100**	
Method	**#Params**	**Accuracy**	**#Params**	**Accuracy**
Teacher	21.3M	0.955	21.3M	0.775
KD [22]	11.2M	0.939	11.2M	0.733
ZSKT [44] (two)	11.2M	0.921	11.2M	0.662
DFAD [13] (two)	11.2M	0.933	11.2M	0.677
Ours(one)	11.2M	**0.934**	11.2M	**0.700**

Table 2: Comparative results of one-stage adversarial knowledge distillation and two-stage one. Here, “one” and “two” denote one-stage and two-stage strategy, respectively.

5. Conclusion

In this paper, we investigate a general one-stage training strategy to enhance the training efficiency of GANs. We categorize GANs into two classes, Symmetric GANs and Asymmetric GANs. For Symmetric GANs, the gradient computation of generator is obtained from the back-propagation of discriminator during the training. The parameters of both generator and discriminator can therefore be updated in one stage. For Asymmetric GANs, we propose a gradient decomposition method, which integrates the asymmetric adversarial losses during forward inference, and decomposes their gradients during back-propagation to separately update generator and discriminator in one stage. We analyze the relation between the above solutions and unify them into our one-stage training strategy. Finally, we computationally analyze the training speedup ratio for OSGANs against TSGANs. Experimental results demonstrate that OSGANs achieve more than 1.5× speedup over TSGANs, and meanwhile preserve the results of TSGANs.

Acknowledgements. This work is supported by National Natural Science Foundation of China (U20B2066, 61976186), Key Research and Development Program of Zhejiang Province (2018C01004), Major Scientifc Research Project of Zhejiang Lab (No. 2019KD0AC01) and Alibaba-Zhejiang University Joint Research Institute of Frontier Technologies.

References

[1] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2017. 3, 6

[2] Shane Barratt and Rishi Sharma. A note on the inception score. *arXiv preprint arXiv:1801.01973*, 2018. 6

[3] Sefi Bell-Kligler, Assaf Shocher, and Michal Irani. Blind super-resolution kernel estimation using an internal-gan. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 284–293, 2019. 2

[4] Mikołaj Bińkowski, Dougal J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. *arXiv preprint arXiv:1801.01401*, 2018. 6

[5] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact: Real-time instance segmentation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 9157–9166, 2019. 2

[6] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *International Conference on Learning Representations (ICLR)*, 2019. 1, 2

[7] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Neural photo editing with introspective adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2017. 2

[8] Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. Mode regularized generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2017. 1, 2

[9] Liang-Chieh Chen, Alexander Hermans, George Papandreou, Florian Schroff, Peng Wang, and Hartwig Adam. Masklab: Instance segmentation by refining object detection with semantic and direction features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4013–4022, 2018. 2

[10] Xinlei Chen, Ross Girshick, Kaiming He, and Piotr Dollár. Tensormask: A foundation for dense object segmentation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2061–2069, 2019. 2

[11] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 379–387, 2016. 2

[12] Jeff Donahue and Karen Simonyan. Large scale adversarial representation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 10542–10552, 2019. 1, 2

[13] Gongfan Fang, Jie Song, Chengchao Shen, Xinchao Wang, Da Chen, and Mingli Song. Data-free adversarial distillation. *arXiv preprint arXiv:1912.11006*, 2019. 1, 7, 8

[14] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning (ICML)*, pages 1180–1189, 2015. 1, 2

[15] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015. 2

[16] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014. 2

[17] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2672–2680, 2014. 1, 2, 3

[18] Shuyang Gu, Jianmin Bao, Hao Yang, Dong Chen, Fang Wen, and Lu Yuan. Mask-guided portrait editing with conditional gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3436–3445, 2019. 2

[19] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (CVPR)*, pages 2961–2969, 2017. 2

[20] Zhenliang He, Wangmeng Zuo, Meina Kan, Shiguang Shan, and Xilin Chen. Attgan: Facial attribute editing by only changing what you want. *IEEE Transactions on Image Processing (TIP)*, 28(11):5464–5478, 2019. 2

[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 *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6626–6637, 2017. 6

[22] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *Advances in Neural Information Processing Systems (NeurIPS)*, 2015. 8

[23] R Devon Hjelm, Athul Paul Jacob, Tong Che, Adam Trischler, Kyunghyun Cho, and Yoshua Bengio. Boundary-seeking generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2018. 6

[24] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics (ToG)*, 36(4):1–14, 2017. 2

[25] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 6

[26] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1125–1134, 2017. 2

[27] Yongcheng Jing, Xiao Liu, Yukang Ding, Xinchao Wang, Errui Ding, Mingli Song, and Shilei Wen. Dynamic instance normalization for arbitrary style transfer. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2020. 2

[28] Yongcheng Jing, Yang Liu, Yezhou Yang, Zunlei Feng, Yizhou Yu, Dacheng Tao, and Mingli Song. Stroke controllable fast style transfer with adaptive receptive fields. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 2

[29] Alexia Jolicoeur-Martineau. The relativistic discriminator: a key element missing from standard gan. In *International Conference on Learning Representations (ICLR)*, 2017. 6

[30] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *International Conference on Learning Representations (ICLR)*, 2018. 1, 2

[31] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4401–4410, 2019. 6

[32] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6

[33] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 734–750, 2018. 2

[34] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4681–4690, 2017. 2

[35] Jae Hyun Lim and Jong Chul Ye. Geometric gan. *arXiv preprint arXiv:1705.02894*, 2017. 6

[36] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *International Journal of Computer Vision (IJCV)*, 128(2):261–318, 2020. 2

[37] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8759–8768, 2018. 2

[38] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 21–37. Springer, 2016. 2

[39] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015. 5

[40] Qi Mao, Hsin-Ying Lee, Hung-Yu Tseng, Siwei Ma, and Ming-Hsuan Yang. Mode seeking generative adversarial networks for diverse image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1429–1437, 2019. 1, 2

[41] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE international Conference on Computer Vision (ICCV)*, pages 2794–2802, 2017. 6

[42] Xudong Mao, Qing Li, Haoran Xie, Raymond Y.K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *International Conference on Computer Vision (ICCV)*, 2017. 3

[43] SC Martin Arjovsky and Leon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2017. 1, 2, 3, 6

[44] Paul Micaelli and Amos J Storkey. Zero-shot knowledge transfer via adversarial belief matching. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 9547–9557, 2019. 1, 7, 8

[45] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 6

[46] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2018. 1, 2, 6

[47] Youssef Mroueh and Tom Sercu. Fisher gan. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2513–2523, 2017. 6

[48] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 271–279, 2016. 2

[49] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2536–2544, 2016. 2

[50] Pedro OO Pinheiro, Ronan Collobert, and Piotr Dollár. Learning to segment object candidates. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1990–1998, 2015. 2

[51] Jiayan Qiu, Yiding Yang, Xinchao Wang, and Dacheng Tao. Hallucinating visual instances in total absentia. In *European Conference on Computer Vision (ECCV)*, 2020. 2

[52] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2015. 6

[53] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. 2

[54] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 91–99, 2015. 2

[55] Mihaela Rosca, Balaji Lakshminarayanan, David Warde-Farley, and Shakir Mohamed. Variational approaches for auto-encoding generative adversarial networks. *arXiv preprint arXiv:1706.04987*, 2017. 6

[56] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3723–3732, 2018. 1

[57] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2234–2242, 2016. 6

[58] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated

recognition, localization and detection using convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2014. 2

[59] Chengchao Shen, Xinchao Wang, Jie Song, Li Sun, and Mingli Song. Amalgamating knowledge towards comprehensive classification. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 3068–3075, 2019. 7

[60] Chengchao Shen, Xinchao Wang, Youtan Yin, Jie Song, Sihui Luo, and Mingli Song. Progressive network grafting for few-shot knowledge distillation. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2021. 1

[61] Chengchao Shen, Mengqi Xue, Xinchao Wang, Jie Song, Li Sun, and Mingli Song. Customizing student networks from heterogeneous teachers via adaptive knowledge amalgamation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019. 7

[62] Jae Woong Soh, Gu Yong Park, Junho Jo, and Nam Ik Cho. Natural and realistic single image super-resolution with explicit natural manifold discrimination. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[63] Hsiao-Yu Fish Tung, Adam W Harley, William Seto, and Katerina Fragkiadaki. Adversarial inverse graphics networks: Learning 2d-to-3d lifting and image-to-image translation from unpaired supervision. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4364–4372. IEEE, 2017. 2

[64] Xintao Wang, Ke Yu, Chao Dong, and Chen Change Loy. Recovering realistic texture in image super-resolution by deep spatial feature transform. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[65] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018. 2

[66] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. Solov2: Dynamic, faster and stronger. *arXiv preprint arXiv:2003.10152*, 2020. 2

[67] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 6

[68] Dingdong Yang, Seunghoon Hong, Yunseok Jang, Tianchen Zhao, and Honglak Lee. Diversity-sensitive conditional generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2019. 1, 2

[69] Yiding Yang, Zunlei Feng, Mingli Song, and Xinchao Wang. Factorizable graph convolutional networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2

[70] Yiding Yang, Jiayan Qiu, Mingli Song, Dacheng Tao, and Xinchao Wang. Distilling knowledge from graph convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[71] Jingwen Ye, Yixin Ji, Xinchao Wang, Xin Gao, and Mingli Song. Data-free knowledge amalgamation via group-stack dual-gan. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1

[72] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. 6

[73] Xiyu Yu, Tongliang Liu, Xinchao Wang, and Dacheng Tao. On compressing deep models by low rank and sparse decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[74] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. In *International Conference on Learning Representations (ICLR)*, 2017. 1, 2

[75] Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. In-domain gan inversion for real image editing. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2020. 2

[76] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 597–613. Springer, 2016. 2

[77] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2223–2232, 2017. 2