UNIVERSITY OF CALIFORNIA
Los Angeles

Towards Fast and Stable GAN via Free Adversarial Training

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Computer Science

by

Jiachen Zhong

2020

ABSTRACT OF THE THESIS

Towards Fast and Stable GAN via Free Adversarial Training

by

Jiachen Zhong
Master of Science in Computer Science
University of California, Los Angeles, 2020
Professor Cho-Jui Hsieh, Chair

State-of-the-art Generative Adversarial Network (GAN) often relies on stabilization methods to stabilize the training by constraining the global Lipschitz continuity. However, the global constraint may result in under-fitting and slow convergence. RobGAN [LH19] proposed a method to control the local Lipschitz value by adversarial training and achieved improved performance and convergence speed. However, the adversarial training procedure in RobGAN leads to significantly increased computational time which makes RobGAN less useful in practice.

In this thesis, we propose to improve the training speed of RobGAN by free adversarial training. In addition, we improve the loss function to diminish the natural flaw of using auxiliary classifier in RobGAN. We evaluate our method on three datasets, CIFAR10, CIFAR100, and IMAGENET-143. Compared with previous works, our methods lead to significant improvements in term of both generation quality and training time in all experiments on these datasets. Moreover, we propose a new point to explain why adversarial training on the discriminator can improve the training of GAN.

The thesis of Jiachen Zhong is approved.

Yizhou Sun

Kai-Wei Chang

Cho-Jui Hsieh, Committee Chair

University of California, Los Angeles

2020

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express my appreciation to my advisor, Prof. Cho-Jui Hsieh, for his guidance and the computational resources supports throughout this research. I would also like to thank Xuanqing Liu for the helpful discussion and feedback on this research. Moreover, I would also like to thank my thesis committee members Prof. Kai-Wei Chang and Prof. Yizhou Sun for their generous assistance. Last but not least, I would like to express my deepest gratitude to my family for their unconditional supports and encouragement in so many ways.

CHAPTER 1

Introduction

The generative ability of machine, which is the ability to learn and reproduce statistically similar data from given data samples, is always an important sign of machine intelligence. In the machine learning field, generative models are often used to perform the generation tasks by directly modeling the data distribution of given data samples. A good algorithm to perform generative tasks is very important not only because it is a critical part of machine intelligence but also it is very useful in many applications. In handling those high dimensional multimedia data like images, a very powerful generative method, Generative Adversarial Network (GAN) [GPM14], has shown considerable success in recent years. Although many research works in GAN have already achieved significant results, lots of problems still remain unsolved. In real practice, GAN training is often very unstable and sensitive to almost every aspect of hyperparameters setting. One factor leads the GAN success is the development of the stabilization techniques, and spectral normalization [MKK18] is the most popular stabilization technique which plays an important role in most of the state-of-the-art image generation works of GAN [MKK18, ZGM18, BDS18]. Spectral normalization constrains the global Lipschitz continuity to force the GAN training become stable, but the global Lipschitz constraints may lead the the model less expressive and consume more time to converge. In real experiments, a complete training on relatively large scale dataset may take several weeks to finish. To solve this problem, instead of constraining the global Lipschitz continuity, RobGAN [LH19] proposed to restrict the local Lipschitz value by using adversarial training [MMS17] during the GAN training to speed up the training. However, using adversarial training may also become a computational bottleneck which leaves the space for further improvement.

Following the footsteps of RobGAN [LH19], we propose a new method by combining the free adversarial training [SNG19] to eliminate the computational bottleneck and improve the training speed. In addition, since the RobGAN uses the auxiliary classifier [OOS17] which may unwittingly lead the model to generate the samples that are easily classified and cause the generated images to lose the diversity inside each class. To solve this problem, we further improve the loss function format to eliminate this problem. Moreover, we conduct the experiments on 3 datasets to verify our proposed method. Our new method consistently outperforms prior works in term of generation quality and training speed in all experiments. We also propose a new perspective of view to explain why adding adversarial training during the training of GAN may help. The contributions can be summarized as follows:

1. We introduce a fast training method of GAN on conditional image generation tasks by combining the free adversarial training [SNG19].

2. We improve the loss format in order to eliminate the disadvantage of using auxiliary classifier.

3. We conduct the experiments on three datasets to verify our method, and perform detail comparison in term of generation quality and training speed with prior strong works [MKK18, LH19].

4. We propose a new perspective to explain why adding adversarial training to the discriminator is able to help the overall GAN training.

CHAPTER 2

Background and Prior Works

In this section, some background information and important prior works of generative models and GAN will be introduced. Moreover, one of the important prior work related to this project, RobGAN [LH19], will be discussed in detail.

2.1 Generative Modeling

In machine learning field, types of models can be roughly divided into two categories, discriminative and generative models. Given some data X with corresponding label Y, discriminative models try to predict the conditional probability $P(Y|X)$ whereas Generative models try to model the joint probability $P(X, Y)$ if with labels, or $P(X)$ if the data are unlabeled. The task of discriminative models can be seen as drawing the decision boundary between data while generative models directly approximate the overall data distribution. Generally speaking, generative tasks is much more difficult than discriminative tasks. Some famous examples of generative models are naïve Bayes, hidden Markov models (HMM), Markova random field (MRF), Gaussian mixture models, auto-encoder, and GAN. Generative models are very useful in a wide variety of applied science and engineering domains. Since the generative models directly approximate the distribution of data, one of the direct applications of generative model is sampling. Another application of generative models is to perform classification by combining with Bayes rules (e.g. naive Bayes, HMM, MRF). Before the recent development of deep generative model (e.g. Variational Auto-Encoder(VAE) [KW13] and GAN) classical generative models are often hard to handle the high-dimension data like images, music, and videos, due to the model capacity. Classical generative models often

can only learn discrete distributions whereas deep generative models could learn continue distributions. With the development of deep learning, deep generative models are able to perform very high fidelity multimedia data generation, which may even not be distinguished by human-being [BDS18].

2.2 Generative Adversarial Network

Generative Adversarial Network (GAN) [GPM14] is a relatively new type of generative model and it can be described as a adversarial game between two sides of computational models, generator(G) and discriminator(D), which are usually neural networks in most of the GAN related works. The game can be simply summarized in the following mathematical formula (2.1):

$$\min_G \max_D \mathop{\mathbb{E}}_{\boldsymbol{x}\sim p_{data}} [\log D(\boldsymbol{x})] + \mathop{\mathbb{E}}_{\boldsymbol{z}\sim p_z} [\log(1 - D(G(\boldsymbol{z})))] \tag{2.1}$$

where G and D are the two players. x are the real training samples from real data distribution p_{data}, and z are the random latent codes from the simple distribution p_z which can usually be directly sampled, like uniform or normal distribution. The G tries to learn a good map between p_z and p_{data} in order to fool the D, and the D tries to distinguish whether the current inputs are real data x or generated(fake) data from G. As D learns a better way to distinguish the real and fake data, G learns a better ability to generate data which are closer to the real data distribution. Therefore, after the training, G can be seen as an approximation of the real data distribution. Moreover, since z is sampled from a simple distribution, G can be efficiently used for sampling. In the real implementation, iteratively updating G and D toward their own optimization goal via mini-batch stochastic gradient descent (SGD) method is a very common way to perform the training of this adversarial competition. Eq. 2.1 gives the most original form of GAN (we use the vanilla GAN to denote it in the later content). The generation quality of this vanilla GAN form is usually limited and its training process is unstable. To conquer these problems in vanilla GAN, many researches have been conducted and they can be roughly divided into three lines: supervision method, architecture exploration, and stabilization technologies which will be introduced in

the following sections.

2.3 Supervision Method of GAN

The vanilla GAN can be seen as an unsupervised learning process since there is no supervised label information involved in the training. By adding supervised label information into GAN training, the generation quality and stabilization of GAN can be improved. In image generation researches, the tasks of utilizing label information are usually called conditional image generation. In term of conditional image generation, there are several ways to utilize label information for both G and D. Conditional GAN (CGAN) [MO14] is the first work which directly feeds the one-hot class vector into both G and D. Later, feeding class information into the hidden layer of networks is also explored in [RAY16]. GAN with auxiliary classifier (ACGAN) [OOS17] utilizes an auxiliary classifier in D to further improve the effectiveness of label information. More recently, conditional GAN with projection discriminator (Projection-cGAN) [MK18] proposes the methods which use conditional batch normalization [DSM17] to handle class information in G and use a learned embedding of classes to project the label information in D. And this combination, conditional batch normalization in G plus projection D , becomes a standard in the later works of large scale image generation tasks [MKK18, ZGM18, BDS18]. BigGAN [BDS18] further explored the ways of utilizing conditional latent code in G by applying the shared embedding of conditional batch normalization and the hierarchy latent code (also called skip-z). Moreover, self-supervised GAN [CZR19] utilizes self-rotation label information to perform the self-supervised learning, and it is successfully applied in the work [LTR19] for large scale semi-supervised conditional image generation.

2.4 Architecture Exploration of GAN

In the vanilla GAN, both G and D are parameterized by fully connected neural networks. This type of neural networks is not efficient and effective in image related tasks. Deep

convolutional GAN (DCGAN) [RMC15] firstly uses the convolutional neural networks to build G and D. Plug and Play Generative Network(PPGN) [NCB17] employs additional pre-trained models to give additional objectives during the training. ProGAN [KAL17] finds training high-resolution GAN can benefit from progressively increasing the resolution during training a single model. In the recent researches of exploring GAN architectures, self-attention GAN (SAGAN) [ZGM18] proposes the non-local (self-attention) blocks to improve the capacity of both G and D which gives the network's ability to modeling long-range dependencies between image regions. Following the result of SAGAN, BigGAN [BDS18] scales up the GAN architectures and finds a large batch size leads a better quality of GAN.

2.5 Stabilization Technologies of GAN

For stabilization technologies, one line is to find the alternative adversarial objective function to enhance the convergence efficiency [ACB17, MLX17, LY17]. Another line for stabilizing GAN training can be seen as a regularization strategy for enhancing the Lipschitz continuity constraint of gradient during training. Wasserstein GAN (WGAN) [ACB17] proposes the concept of Wasserstein distance which is a better alternative of the original probability format loss. By using empirical weight clipping, WGAN is able to achieve a more stable training. Following WGAN, WGAN with gradient penalty (WGAN-GP) [GAA17] further improves the stability by using an improved gradient penalty method. Many other methods, weight normalization [SK16], singular value clipping [SMS17], orthogonal regularization [BLR16], orthogonal normalization [HLL18], also show benefits to improve the stability of training. To the state-of-the-art regularization method, spectral normalization [MKK18] controls the global Lipschitz continuity of networks by dynamically normalizing the parameters in each layer with running estimates of first singular values, and it has been successfully applied in large scale image generation tasks [MKK18, ZGM18, BDS18]. Instead of constraining global Lipschitz continuity, RobGAN [LH19] proposes the methods which utilize adversarial training [MMS17] to maintain the local Lipschitz value in order to lead a fast as well as better convergence. Moreover, empirically, GAN training is relatively more stable under the

imbalanced updates between G and D and this trick is used in training of many cutting edge GAN models [MKK18, MK18, ZGM18, BDS18]. However, imbalanced updates trick may cause extra computational costs. Two time-scale update rule (TTUR) [HRU17] uses different learning rates between G and D to perform the balance and avoid repeated computational costs. In practice, these two methods are usually used together.

2.6 RobGAN

RobGAN is a recently proposed method of training conditional GAN in image generation tasks. Without using methods to regularize the global Lipschitz continuity, RobGAN empirically finds that constraining the local Lipschitz values by adding adversarial training [MMS17] in the training of GAN is able to speed up the overall convergence and improve the generation quality. In order to perform the standard adversarial training, RobGAN uses the auxiliary classifier [OOS17] in D to handle the label information. One drawback of RobGAN is the computational bottleneck of the adversarial training. In this project, we use the idea of free adversarial training [SNG19] to further reduce the computational cost in the RobGAN training. Another problem of the RobGAN is the natural flaw of using the auxiliary classifier which may lead the intra-class model collapse. We firstly introduce the notations and review the training process of the original RobGAN in Alg. 1 and then extend it to the "fast" version in Chapter 3.1.

We firstly introduce the notations. We use X and Y to present the set of training samples and labels; x (x^f for fake, and x^r for real), y and z to represent the mini-batch of samples, labels, and latent noise sampled by function $Batch(\cdot)$; G and D to represent the generator and discriminator parameterized by θ_G and θ_D; $step_D$ to represent the number of updates of the D per one update of the G; $N(\cdot)$ to represent the function to generate latent noise z and labels y^f for generated fake images; σ, ϵ, τ, and K to represent the perturbation, perturbation bound, step size, and steps of L_∞ PGD-attack [MMS17]; g to represent the gradient; $\mathcal{L}$ to represent a general loss function for simplicity (i.e. $\mathcal{L}$ could be different under different cases); η to represent the learning rate; $U(\cdot)$ to represent the uniform distribution.

Algorithm 1: RobGAN Training

1 **for** *iterations* **do**
2 **for** $step_D$ **do**
3 $z, y^f \leftarrow Batch(N(\cdot))$
4 $x^f \leftarrow G(z)$
5 $x^r, y^r \leftarrow Batch(X, Y)$
6 $\sigma \leftarrow U(-\epsilon, \epsilon)$
7 **for** K **do**
8 $g_{adv} \leftarrow \nabla_{(x^r+\sigma)} \mathcal{L}[x^r + \sigma, y^r, \theta_D]$
9 $\sigma \leftarrow \sigma + \tau \cdot sign(g_{adv})$
10 $\sigma \leftarrow clip(\sigma, -\epsilon, \epsilon)$
11 **end**
12 $g^r_{\theta_D} \leftarrow \nabla_{\theta_D} \mathcal{L}[x^r + \sigma, y^r, \theta_D]$
13 $g^f_{\theta_D} \leftarrow \nabla_{\theta_D} \mathcal{L}[x^f, y^f, \theta_D]$
14 $\theta_D \leftarrow \theta_D - \eta_D \cdot (g^r_{\theta_D} + g^f_{\theta_D})$
15 **end**
16 $z, y^f \leftarrow Batch(N(\cdot))$
17 $x^f \leftarrow G(z)$
18 $g_{\theta_G} \leftarrow \nabla_{\theta_G} \mathcal{L}[x^f, y^f, \theta_D, \theta_G]$
19 $\theta_G \leftarrow \theta_G - \eta_G \cdot g_{\theta_G}$
20 **end**

As shown in the Alg. 1, the major difference compared with other GAN training as well as the computational bottleneck in RobGAN is the K-step PGD-attack (row 7 to 10 in Alg. 1). It requires K steps more forward and backward processes of D to construct the perturbation σ. In the experiments of the RobGAN [LH19], $step_D$ and K are both set to 5, which costs 25 forward processes of D to build the perturbation in 1 iteration. Another key improvement of RobGAN is the new format of loss function which can be summarized in the following

Eq. 2.2:

$$\begin{aligned}\mathcal{L}_G &= -\mathbb{E}[\log \Pr(real|D^b(x^f))] - \mathbb{E}[\log \Pr(y^f|D^c(x^f))] \\ \mathcal{L}_D &= \mathcal{L}_D^r + \mathcal{L}_D^f\end{aligned} \tag{2.2}$$

where $\mathcal{L}_G$ and $\mathcal{L}_D$ are the loss function of G and D to minimize; D^b and D^c are the outputs of D for performing fake/real adversarial classification and multi-classes classification (auxiliary classifier). $\mathcal{L}_D^r$ and $\mathcal{L}_D^f$ are the loss function of D for real and fake data respectively. The $\mathcal{L}_G$ is same as the ACGAN [OOS17] and it aims to force the G to generate data close to real samples conditionally. For $\mathcal{L}_D^r$ and $\mathcal{L}_D^f$:

$$\begin{aligned}\mathcal{L}_D^r &= -\mathbb{E}[\log \Pr(real|D^b(x^r + \sigma)) - \mathbb{E}[\log \Pr(y^r|D^c(x^r + \sigma))] \\ \mathcal{L}_D^f &= -\mathbb{E}[\log \Pr(fake|D^b(x^f))]\end{aligned} \tag{2.3}$$

where σ is the perturbation. Same as ACGAN, $\mathcal{L}_D^r$ aims to push the D to recognize the real data and predict its class labels. The major change of RobGAN from ACGAN loss is $\mathcal{L}_D^f$ which only force D to distinguish the fake data without classifying them.

2.7 Evaluation Metrics of GAN

Different from discriminative tasks (e.g. classification or regression), it is hard to numerically measure the quality of data generation, especially for high dimensional data like images. A very direct way is to use human eyes for scoring the generated images but it is very expensive, subjective, and hard to be applied to large amounts of data. Besides using human force to evaluate, an idea is to use a pre-trained powerful computational model to measure the quality. Currently, there are two popular and reliable methods which use pre-trained Inception-V3 Network [SVI16]: Inception Scores (IS) [SGZ16] and Fréchet Inception Distance (FID) [HRU17] which will be used in this project for evaluation methods. Both of the methods utilize the statistic of the Inception Network's outputs to perform evaluation and they can be efficiently computed. IS aims to measure the realism and diversity of the images. It can be described as the following mathematical form:

$$IS = \exp(\mathop{\mathbb{E}}_{\boldsymbol{x}\sim p_g}[D_{KL}(p(y|x)||p(x))]) \tag{2.4}$$

where D_{KL} is the KL-divergence, p_g is the generated distribution, $p(y|x)$ and $p(x)$ are the conditional and marginal label distribution of generated images samples measured by pre-trained Inception Network. The higher the IS score is, the better the generated images are. To our best knowledge, IS scores for real datasets are always higher than the generated data in all works so far, no research work on image generation currently can match or outperform real datasets on IS metric. However, IS is limited to measure the sample diversity inside each class (e.g. IS should still be high even if a model can only produce one good sample for each class).

FID mainly measures the difference between the real data distribution and generated data distribution by Fréchet distance [DL82] . The mathematical form is described as following:

$$FID = ||\mu_r - \mu_g||^2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{\frac{1}{2}}) \tag{2.5}$$

where μ_r, μ_g, Σ_r, Σ_g are the respective means and covariance matrices of the 2048-dimensional activations of the Inception-V3 pool3 layer of real and generated samples. The lower the FID score is, the better the generated images are. If the two sets of images are the same, FID should be zero. Moreover, FID is able to measure the intra-class diversity.

In real practice, FID and IS are usually used together to measure the generation quality of GAN. And they are often processed by sampling large enough numbers of generated data (usually 50K) with taking the average of computing multiple times. In this project, we use both FID and IS as the metric to evaluate the generation quality.

CHAPTER 3

Proposed Methods

In this section, we introduce the proposed method which overcomes the disadvantages of RobGAN and we call it **Fast-RobGAN** for simplicity in the following content. We firstly discuss the Fast-RobGAN on how to incorporate free adversarial training in RobGAN to eliminate the computational overhead. Then, we propose an improved loss function format modified from RobGAN in Chapter 3.2. In addition, we introduce a new perspective about the theories of global and local Lipschitz continuity as well as their relations to adversarial training in Chapter 3.3 in order to better explain the effectiveness of adversarial training in GAN.

3.1 Fast-RobGAN Training

To solve the disadvantage of using adversarial training in RobGAN, we use the idea of free adversarial training [SNG19]. The detail of Fast-RobGAN training is shown in the Alg. 2. Most notations in the Alg. 2 are same in Alg. 1 but we change the step of L_∞ PGD-attack K to the free-step M, and change the perturbation σ to δ as the global perturbation. The main difference between the original version and "fast" version of RobGAN training is that we use a global perturbation δ to accumulate the perturbation value during the whole training process, and update D as well as δ in every Free step M (row 7 to 13 in Alg. 2) which can be computed by using only 1 forward and backward process. By taking the advantage of free adversarial training [SNG19], we could perform a more frequent update of D with perturbation and speed up the convergence of overall training. One limitation of using free adversarial training is that we have to perform M updates of D in order to simulate the K

steps PGD-attack and this will lead the unbalanced updates between D and G. Although, most prior works [MKK18, GAA17, BDS18] use more frequent update of D than G during training, it is necessary to search a new hyperparameters setting of $step_D$ and M in order to balance the training. In real practice, setting $step_D = 1$ and $M = 2$ are empirically good for most of the experiments. The experiment results show the effectiveness of the Fast-RobGAN training in both generation quality and convergence speed and the detail will be presented in Chapter 5.

Algorithm 2: Fast-RobGAN Training

$\delta \leftarrow U(-\epsilon, \epsilon)$
for *iterations* **do**
  for $step_D$ **do**
    $z, y^f \leftarrow Batch(N(\cdot))$
    $x^f \leftarrow G(z)$
    $x^r, y^r \leftarrow Batch(X, Y)$
    for M **do**
      $g_{adv}, g^r_{\theta_D} \leftarrow \nabla_{(x^r+\delta),\theta_D} \mathcal{L}[x^r + \delta, y^r, \theta_D]$
      $g^f_{\theta_D} \leftarrow \nabla_{\theta_D} \mathcal{L}[x^f, y^f, \theta_D]$
      $\theta_D \leftarrow \theta_D - \eta_D \cdot (g^r_{\theta_D} + g^f_{\theta_D})$
      $\delta \leftarrow \delta + \tau \cdot sign(g_{adv})$
      $\delta \leftarrow clip(\delta, -\epsilon, \epsilon)$
    end
  end
  $z, y^f \leftarrow Batch(N(\cdot))$
  $x^f \leftarrow G(z)$
  $g_{\theta_G} \leftarrow \nabla_{\theta_G} \mathcal{L}[x^f, y^f, \theta_D, \theta_G]$
  $\theta_G \leftarrow \theta_G - \eta_G \cdot g_{\theta_G}$
end

3.2 Improved Loss Function

Besides the change of the training process, we further modify the loss function format from RobGAN. Overall, we use the similar loss function format as described in Eq. 2.2 and 2.3 but we change fake/real adversarial part of loss function form probability format to hinge format [LY17, MKK18]. This hinge format loss function can be seen as a type of loss to measure Wasserstein distance (also called Earth-Mover distance) [ACB17] bounded by hinge margin which reduces the gradient vanishes and instability problem in the original probability adversarial loss format.

Fast-RobGAN still inherits the auxiliary classifier for handling label information from RobGAN because it is more suitable for adversarial training. However, one natural flaw of using auxiliary classifier as reported in [OOS17, MK18] is that the G is very often to unexpectedly generate samples which are easy to be classified by the auxiliary classifier. This phenomenon can also be described as the model collapse inside each class (intra-class model collapse) in which the G can only produce almost one similar sample inside each class. Under this situation, the samples from G hold diversity among classes but not inside classes. In addition, this problem has a tendency to be worse as the number of classes increases. In term of measurement metrics (IS and FID), the model suffered from this problem tends to generate relatively good IS score but with limited FID value. In RobGAN, this problem is mitigated by removing the classification part in $\mathcal{L}_D^f$ as described in Eq. 2.3. To further conquer this problem, instead of doing nothing to the generated samples during the updates of D as in RobGAN, we try to minimize KL-divergence between class conditional probability of generated samples $\Pr(y^f|D^c(x^f))$ and a uniform distribution $U(\cdot)$ among all classes, which makes the generated samples not easy to be classified by the auxiliary classifier. Moreover, we introduce a coefficient to the classification part of the loss of G in order to control this problem. The overall new loss format of Fast-RobGAN can be formulated as:

$$\begin{aligned} \mathcal{L}_G &= -\mathbb{E}[D^b(x^f)] - \alpha_c^g \cdot \mathbb{E}[\log \Pr(y^f|D^c(x^f))] \\ \mathcal{L}_D &= \mathcal{L}_D^r + \mathcal{L}_D^f \end{aligned} \tag{3.1}$$

where $\mathcal{L}_D^r$ and $\mathcal{L}_D^f$ are:

$$\begin{aligned}\mathcal{L}_D^r &= \mathbb{E}[max(0, 1 - D^b(x^r + \delta))] - \mathbb{E}[\log \Pr(y^r | D^c(x^r + \delta))] \\ \mathcal{L}_D^f &= \mathbb{E}[max(0, 1 + D^b(x^f))] + \mathbb{E}[KL(\Pr(y^f | D^c(x^f), U(\cdot))]\end{aligned} \tag{3.2}$$

where α_c^g is the coefficient to control the classification part G, δ is the global perturbation in free adversarial training, $KL(\cdot)$ is the KL-divergence, $U(\cdot)$ is the uniform distribution on the classes, $D^b(\cdot)$ and $D^c(\cdot)$ are the output of adversarial part and auxiliary classifier part of $D(\cdot)$. In our experiments, we empirically find that the α_c^g is often needed to be decreased if the number of classes of the datasets increases.

3.3 Global v.s. Local Lipschitz Continuity

In this section, we provide a new perspective of view to explain why adding adversarial training to discriminator could help the overall convergence. Generally speaking, the adversarial training can be seen as a simulation of forcing local Lipschitz continuity which leads the stability at the late stage of training but not restricts the convergence speed at the early stage of training. We firstly start with global Lipschitz continuity. Then we introduce the local Lipschitz continuity and draw its relationship to the adversarial training.

Constraining global Lipschitz continuity as spectral normalization [MKK18] should provide more stability of training but it may lead the model to converge slow and express poorly. Let's consider the following equations:

$$\frac{\mathcal{M}\{\mathcal{L}[D(G(z)), y], \mathcal{L}[D(x), y]\}}{||G(z) - x||} \approx \mathcal{M}'(\cdot) \cdot \mathcal{L}'(\cdot) \cdot D'(\cdot) \cdot G'(z) \tag{3.3}$$

where $\mathcal{M}$ is a metric space (e.g. Euclidean Distance), $||\cdot||$ represents some types of norm (e.g L_∞), $\mathcal{L}$ is the loss function, $D(\cdot)$ here is a fixed Discriminator at some training moment, z, x and y are the latent noise, real data samples, and labels. Here, we use a general format of metric space $\mathcal{M}$ and loss function $\mathcal{L}$ which means different GAN methods have various types. Eq. 3.3 actually approximates the gradient of updating $G(\cdot)$ using finite difference method. If $D(\cdot)$ holds the global Lipschitz continuity, then the following Eq. 3.4 should hold

for any $||x' - x||$ where $x' \neq x$ within the input space:

$$\frac{\mathcal{M}\{\mathcal{L}[D(x'), y], \mathcal{L}[D(x), y]\}}{||x' - x||} \leq \mathcal{C}_{\mathcal{D}} \tag{3.4}$$

where $\mathcal{C}_{\mathcal{D}}$ is the global Lipschitz constant. Therefore, Eq. 3.3 should be bounded by some constant $\mathcal{C}_{\mathcal{D}}$ under the global Lipschitz continuity constraint:

$$\frac{\mathcal{M}\{\mathcal{L}[D(G(z)), y], \mathcal{L}[D(x), y]\}}{||G(z) - x||} \approx \mathcal{M}'(\cdot) \cdot \mathcal{L}'(\cdot) \cdot D'(\cdot) \cdot G'(z) \lessapprox \mathcal{C}_{\mathcal{D}} \tag{3.5}$$

This property will lead stability but also bound the gradient for updating $G(\cdot)$ during the whole training process which may lead the convergence slow. It will be better if we could produce this constraint when the $G(\cdot)$ is only close to the real training samples, which could be formulated as local Lipschitz continuity as following:

$$\frac{\mathcal{M}\{\mathcal{L}[D(x + \Delta), y], \mathcal{L}[D(x), y]\}}{||\Delta||} \leq \widetilde{\mathcal{C}_{\mathcal{D}}} \tag{3.6}$$

where Δ is a small difference, x is the training samples (instead of all points within the input space), and $\widetilde{\mathcal{C}_{\mathcal{D}}}$ is the local Lipschitz constant. Under this situation, Eq. 3.3 can only be bounded by some constant $\widetilde{\mathcal{C}_{\mathcal{D}}}$ when $||G(z) - x|| \leq \Delta$. When $||G(z) - x|| > \Delta$, there is no limitation of the gradient for updating $G(\cdot)$. However, this smarter constraint is hard to be applied in real practice but we can simulate this local Lipschitz continuity constraint by using the adversarial training which can be described as:

$$\min_{D} \mathcal{L}[D(x + \sigma), y] \tag{3.7}$$

where σ is a small perturbation generated from PGD-attack [MMS17] which is usually bounded by some predefined value (i.e. $||\sigma|| \leq \sigma_{max}$) , y is the label information, and $\mathcal{L}$ the loss function. If we have an good $D(\cdot)$ which means the $\mathcal{L}[D(x), y] \approx 0$, then Eq. 3.7 can be seen as:

$$\min_{D} \mathcal{M}\{\mathcal{L}[D(x + \sigma), y], \mathcal{L}[D(x), y]\} \approx \min_{D} |\mathcal{L}[D(x + \sigma), y] - 0| = \min_{D} \mathcal{L}[D(x + \sigma), y] \tag{3.8}$$

where $\mathcal{M}$ are specifically selected as the 1-D Euclidean Distance here. And since σ is a constant for the update of D during adversarial training, though not common, Eq. 3.8 can

also be proportionally described as following:

$$\min_{D} \frac{\mathcal{M}\{\mathcal{L}[D(x+\sigma), y], \mathcal{L}[D(x), y]\}}{||\sigma||} \tag{3.9}$$

which can be seen as a simulation of building local Lipschitz continuity described in Eq. 3.6 which makes D more robust around training data points. However, this format only considers the situation in which σ can only lead the $\mathcal{L}(\cdot)$ increase (i.e $\forall\sigma : ||\sigma|| \leq \sigma_{max}, \mathcal{L}[D(x), y] \leq \mathcal{L}[D(x+\sigma), y]$). Strictly speaking, if there is no extra condition, σ should be possible to lead the $\mathcal{L}(\cdot)$ decrease as well and this unbounded decrease may also cause an unwanted gradient during the training which leads the training unstable. However, since the major objective of the training process is trying to minimize the objective: $\mathcal{L}[D(x), y]$, a good trained $D(\cdot)$ should have a relatively low loss value on the training samples, which means there should only be a little σ which may only lead a marginally decrease around the training points. Therefore, if there is a good $D(\cdot)$ at the late stage of training, $G(\cdot)$ can still enjoy the stability benefited from this method.

In another point of view, if we can get a perfect $D(\cdot)$ and perform a perfect Adversarial Defense :

$$\forall\sigma : ||\sigma|| \leq \sigma_{max}, \mathcal{M}\{\mathcal{L}[D(x+\sigma), y], \mathcal{L}[D(x), y]\} = 0, \mathcal{L}[D(x), y] \leq \mathcal{L}[D(x+\sigma), y] \tag{3.10}$$

which means there is no difference between $\mathcal{L}[D(x+\sigma), y]$ and $\mathcal{L}[D(x), y]$ under metric $\mathcal{M}$. Under this situation, $G(\cdot)$ cannot real force $G(z)$ to approach x when $||G(z) - x|| \leq \sigma_{max}$ because $G(\cdot)$ is unable to receive the information of difference between $G(z)$ and x via $D(\cdot)$, which means the gradient for updating $G(\cdot)$ is zero (i.e. $\widetilde{\mathcal{C}_{\mathcal{D}}} = 0$).

In real practice, adversarial training can not guarantee a strict local Lipschitz continuity constraint and it is almost impossible to perform perfect Adversarial Defense. Its performances are various on different datasets but it can make sure that the gradient for updating $G(\cdot)$ can progressively decrease as the $G(z)$ reaches x. This will lead the training stable at the late stage and make the model easier to converge to an optimal solution.

CHAPTER 4

Experiments

4.1 Datasets

We test our methods on 3 datasets: CIFAR10, CIFAR100 [Kri09], and IMAGENET-143. CIFAR10 and CIFAR100 both contain 50,000 32×32 (32px) resolution RGB images evenly distributed in 10 and 100 classes respectively. Moreover, we use the IMAGENET-143 dataset, which is a subset of IMAGENET [RDS15], used in SNGAN [MKK18] and RobGAN [LH19] for relatively larger scale experiments. IMAGENET-143 contains 180,373 images in 143 classes and the experiments focus on both 64×64 (64px) and 128×128 (128px) resolutions in order to compare with previous works.

4.2 GAN Architectures in Experiments

In order to perform a better comparison, we use the ResNet based architectures used in the previous works [GAA17, MKK18, LH19] to perform the experiments. However, we replace any other label handling methods (e.g. projection embedding in SNGAN architecture) with auxiliary classifier for all Fast-RobGAN experiments. For **CIFAR10** and **CIFAR100**, we use the same architecture for both two datasets which is used in WGAN-GP [GAA17], SNGAN [MKK18], and Projection-cGAN [MK18]. We do not perform the experiments which increase the number of feature maps in G or D as SNGAN but we keep the number of feature maps equal to 128 in both G and D for all CIFAR experiments for comparison. For **IMAGENET-143**, we use the same network architecture used in SNGAN and RobGAN on this dataset to perform the experiments on both 64px and 128px resolutions.

Table 4.1: Hyperparameters settings of Fast-RobGAN training

	iter.	*B.S.*	$step_D$	B, I	M	ϵ/τ	α_c^g
CIFAR10	240k	64	1	0.5, 80k	2	2e-2	1.0
CIFAR100	240k	64	1	0.5, 80k	2	2e-2	0.2
IMAGENET-143 (64px)	120k	64	1	0.5, 30k	2	3e-2	1.0
IMAGENET-143 (128px)	120k	64	1	0.5, 30k	2	3e-2	1.0

note: *iter.* is the total iterations, *B.S.* is the batch size

Besides using the conditional batch normalization [DSM17] in G as prior works [MKK18, ZGM18, LH19, BDS18], we do not use any other reparametrization or regularization methods (e.g. spectrum normalization or gradient penalty) in either G or D for all Fast-RobGAN experiments.

4.3 Hyperparameters and Empirical Settings

We use Adam Optimizer [KB14] with initial learning rate 0.0002 for both D and G ($\eta_D^{init}/\eta_G^{init} = 0.0002$) in all experiments. We set Adam Optimizer $\beta_1 = 0.0$ and $\beta_1 = 0.9$ for all experiments. For learning rate schedule, unlike previous works, we empirically find using exponential learning rate anneal to progressively decrease the learning rate during training can improve the stability and generation quality. This learning rate schedule can be described as:

$$\eta = \eta^{init} \cdot B^{\frac{i}{I}} \tag{4.1}$$

where η is the current learning rate, η^{init} is the initial learning rate, i is the number of iterations, B is the exponential base, and I is the denominator of the exponential index. B and I are two hyperparameters which can be changed in different experiments. We find the advantage of this annealing method on CIFAR10 and IMAGENET-143 datasets. However, since this method has two hyperparameters to tune and one of the drawbacks of this method is that it is hard to find a good setting on large scale experiments. The hyperparameters settings of Fast-RobGAN training are summarized in Tab. 4.3.

CHAPTER 5

Results

5.1 CIFAR10

We report the generation quality measurement in IS and FID of CIFAR10 in Tab. 5.1. Our Fast-RobGAN is able to achieve better results compare with spectral normalization GAN (SNGAN) and RobGAN. We reproduce the SNGAN experiments according to the paper [MKK18, MK18] and official code[1]. Since the $step_D$ in original SNGAN setting is 5 which is computational heavy, we also try the experiments which set the $step_D = 1$ and use the SAGAN setting [ZGM18] ($step_D = 1$, spectrum normalization in both D and G, $\eta_D/\eta_G = 4e-4/1e-4$). However, two settings collapse at the early training stage, thus, we report the N/A in Tab. 5.1. The original SNGAN setting uses the Projection Embedding (denoted as SNGAN(Projection)) to handle the label information. For comparison, we also try the experiments of using auxiliary classifier (denoted as SNGAN(AC)) as done in [GAA17, MK18, MK18]. For RobGAN, there is no experiments on CIFAR10 from the original work [LH19]. We try to perform CIFAR10 experiments on RobGAN setting but we are unable to achieve a comparable result with SNGAN and Fast-RobGAN. We still report the best RobGAN result we can achieve within 25k seconds training time in Tab. 5.1.

For the experiments of Fast-RobGAN, besides the optimal setting (row 7 in Tab. 5.1), we also report some additional settings which remove one part of our proposed new modification from our optimal setting to perform comparisons. We firstly try the to use the original RobGAN loss format as decribed in Eq. 2.2, 2.3, denoted as Fast-RobGAN (Eq. 2.2, 2.3) in Tab. 5.1 but the training collapses at the early stage. We also try the settings which

[1] `https://github.com/pfnet-research/sngan_projection`

Table 5.1: Experimental results of CIFAR10

	IS	FID	Time(sec.)
real data	11.24±.10	5.30	N/A
SNGAN(Projection)	7.47±.13	14.59	24.5k
SNGAN(AC)	7.48±.22	15.38	24.5k
SNGAN($step_D = 1$)	N/A	N/A	N/A
SNGAN(SAGAN setting)	N/A	N/A	N/A
RobGAN	6.95±.06	25.75	25k
Fast-RobGAN	**7.82±0.09**	**12.77**	**22.8k**
Fast-RobGAN(Eq. 2.2, 2.3)	N/A	N/A	N/A
Fast-RobGAN($\epsilon = 0$)	**7.79±0.12**	14.43	**22.8k**
Fast-RobGAN(w/o anneal)	**8.09±0.11**	14.67	**22.8k**
Fast-RobGAN(w/o KL-term)	7.37±0.17	16.26	**22.8k**

note: IS and FID are measured by 50k and 5k generated samples respectively. N/A denotes the experiment collapses at the early training stage or is unable to achieve a comparable result.

remove the adversarial training, denoted as Fast-RobGAN($\epsilon = 0$) in Tab. 5.1, without using exponential learning rate anneal, denoted as Fast-RobGAN(w/o anneal) in Tab. 5.1, and without using the new proposed KL-divergence term, denoted as Fast-RobGAN(w/o KL-term) in Tab. 5.1. All of these settings receive a obvious deterioration on FID scores which show the effectiveness of our new proposed methods.

To compare the convergence speed, our Fast-RobGAN is faster than SNGAN and RobGAN with a better performance. As shown in Fig. 5.1, Fast-RobGAN demonstrates the acceleration compare with SNGAN especially at the early stage of the training which gives the experimental supports to our assumption proposed in Chapter 3.3.

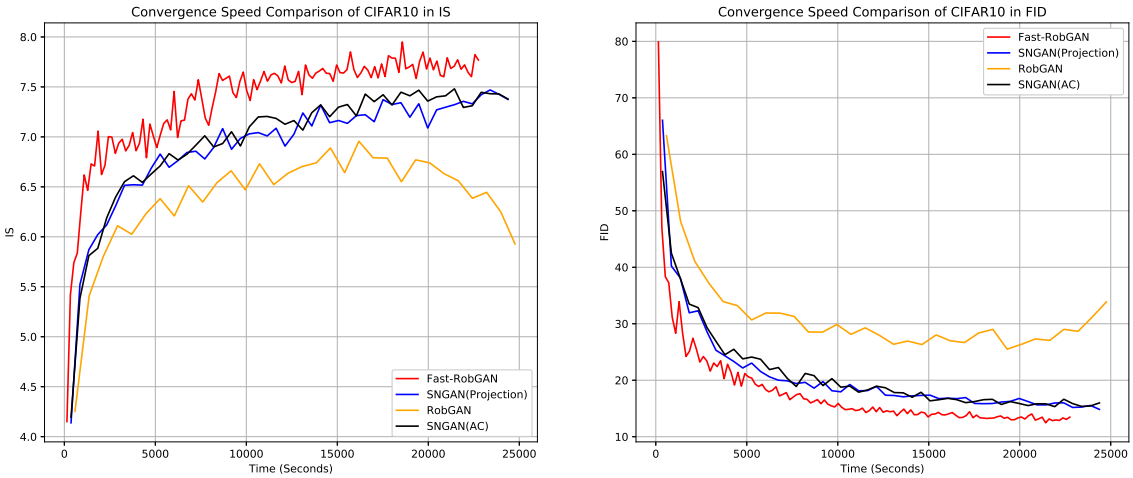

Figure 5.1: Convergence speed of CIFAR10

5.2 CIFAR100

For CIFAR100, we try the experiments with similar settings as used in CIFAR10. We directly use the CIFAR10 hyperparameters for SNGAN and RobGAN experiments. We report the generation quality in term of IS and FID scores in Tab. 5.2. In the experiments of CIFAR100 which has a relatively more number of classes, the disadvantage of using auxiliary classifier exposes. SNGAN with auxiliary classifier, SNGAN(AC), collapses at the early stage of the training. We sample the images from SNGAN(AC) (row 4 in Fig. A.2) and find it suffers from the serious intra-class models collapse. RobGAN is able to continue training but the inconsistency between IS and FID scores also indicates the occurrence of model collapse inside each class. The image samples of RobGAN (row 5 in Fig. A.2) also supports the occurrence of this problem. Therefore, we tune down the α_c^g to 0.2 in the Fast-RobGAN and our method is able to considerably outperform the SNGAN and RobGAN in both IS and FID metrics. We also try the experiments of removing one part from the optimal settings (row 8, 9, 10, 11 in Tab. 5.2) to perform the comparison. Similar to the result from CIFAR10 experiments, model collapses if using the original RobGAN loss (Eq. 2.2, 2.3). Also, removing any parts from the optimal setting may lead the performance slightly increase on IS score but gives much worse FID. Moreover, if we remove the KL-divergence term (row 11 in Tab. 5.2), the

Table 5.2: Experimental results of CIFAR100

	IS	FID	Time(sec.)
real data	14.79±0.15	5.91	N/A
SNGAN(Projection)	7.86±0.21	18.29	25.6k
SNGAN(AC)	N/A	N/A	N/A
SNGAN($step_D = 1$)	N/A	N/A	N/A
SNGAN(SAGAN setting)	N/A	N/A	N/A
RobGAN	7.24±0.21	36.27	26k
Fast-RobGAN	**8.87±0.06**	**17.27**	**23.1k**
Fast-RobGAN(Eq. 2.2, 2.3)	N/A	N/A	N/A
Fast-RobGAN($\epsilon = 0$)	**8.38±0.21**	19.43	**23.1k**
Fast-RobGAN(w/o anneal)	**8.95±0.31**	19.53	**23.1k**
Fast-RobGAN(w/o KL-term)	7.42±0.18	24.02	**23.1k**

note: IS and FID are measured by 50k and 5k generated samples respectively. N/A denotes the experiment collapses at the early training stage or is unable to achieve a comparable result.

Fast-RobGAN will receive an obvious performance drop which indicates the effectiveness of adding the KL-divergence term. In term of computational time, the Fast-RobGAN is still able to use slightly less training time to achieve a better generation quality compare with SNGAN and RobGAN which can be directly observed in Fig. 5.2.

To further study the effectiveness of our method on intra-class model collapse problem, we trained the Fast-RobGAN with different α_c^g and perform the random conditional sampling at the same iterations (120k). We present the samples in Fig. 5.3. The α_c^g control the conditioning capacity of the G during the training, thus, a larger α_c^g may lead the auxiliary classifier of G over-fitting which causes the intra-class model collapse and a smaller α_c^g may cause the G loss the ability of conditioning. As shown in the Fig. 5.3, when the α_c^g is very big ($\alpha_c^g = 2.0$) the model can only generate almost one similar sample with limited variety (in term of shapes, colors, etc.) inside each class. As the α_c^g decreases, G is able to produce more

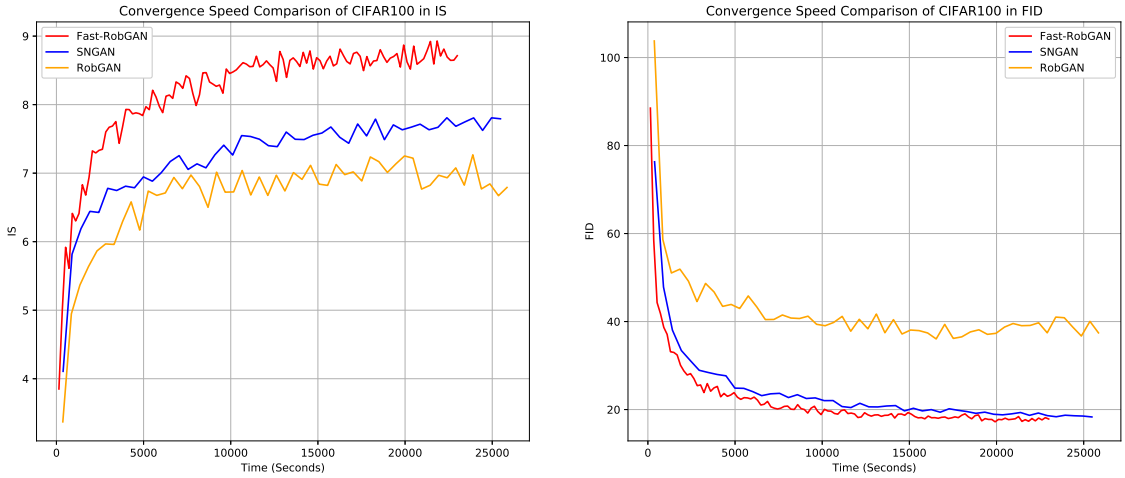

Figure 5.2: Convergence speed of CIFAR100

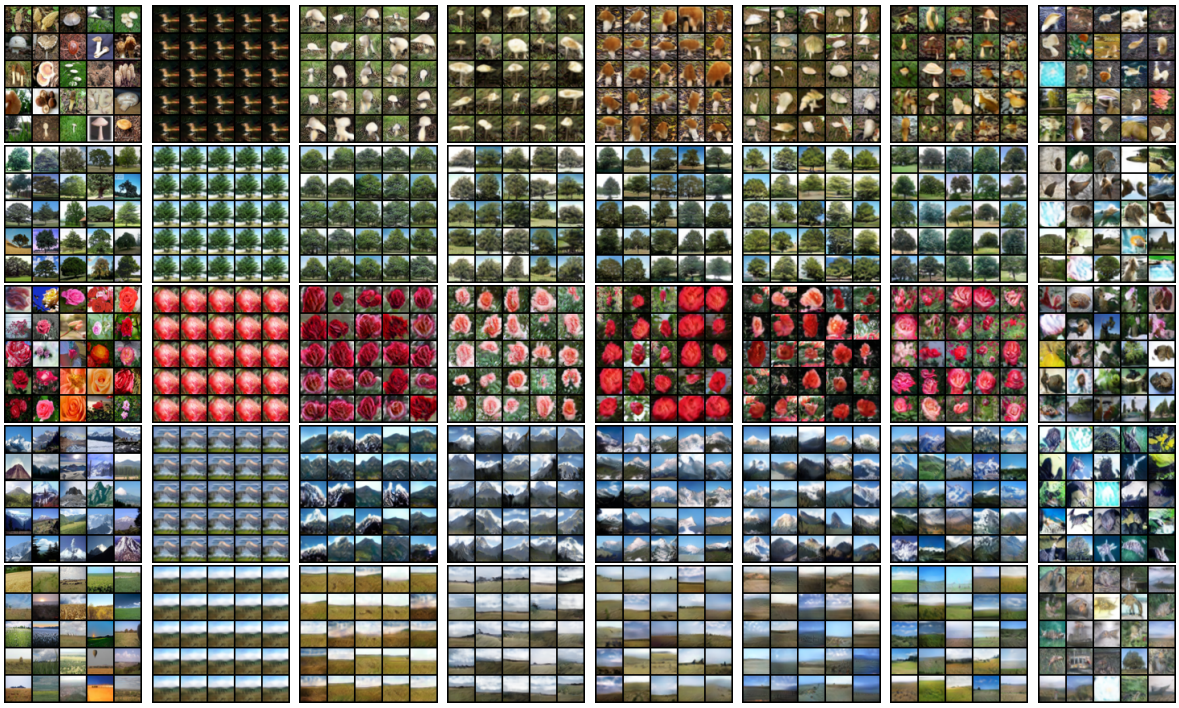

Figure 5.3: Random image samples of 5 classes from CIFAR100. From top to down, each row is a class from CIFAR100. From left to right, each column are sampled from Real Data, and Fast-RobGAN trained with $\alpha_c^g = 2.0, 1.0, 0.8, 0.6, 0.4, 0.2, 0.1$ at 120k iterations.

various samples insides classes. However, when the α_c^g is relatively small ($\alpha_c^g = 0.1$), the G starts to lose the ability to produce the correct samples corresponding to class information. The effectiveness of the α_c^g seems not linear and various for different classes. Therefore, a suitable α_c^g is necessary for the good training of Fast-RobGAN.

5.3 IMAGENET-143

We perform experiments on both 64px and 128px resolutions. We also reproduce experiments of SNGAN and RobGAN according to the official released codes and hyperparameters settings[1][2]. For those reproduction experiments, we can achieve a similar metric results as reported in the original papers [MKK18, LH19].

Table 5.3: Experimental results of IMAGENET-143 (64px)

	IS	FID	Times(sec.)
real data	27.9±0.419	0.48	N/A
SNGAN	10.7±0.17(11.5‡)	29.7	220k
RobGAN	24.62±0.31(∼22.5⋆)	14.64	180k
Fast-RobGAN	**22.23±0.35**	**12.04**	**36k**
Fast-RobGAN(Eq. 2.2, 2.3)	**25.49±0.29**	**14.03**	**36k**
Fast-RobGAN(w/o KL-term)	**25.61±0.47**	**13.94**	**36k**

note: ‡, ⋆ reported in 1, [LH19]. IS and FID are measured by 50k generated samples.

For 64px resolution, we present our result in Tab. 5.3. Fast-RobGAN can easily beat SNGAN under both IS and FID metrics and achieve similar performance as RobGAN. In term of computational time, as shown in Fig. 5.4, Fast-RobGAN is very cheap which only spends about **17%** and **20%** overall time of SNGAN and RobGAN respectively to achieve a substantially better generation quality. In term of IS score, Fast-RobGAN achieves a slightly lower performance than the RobGAN but the FID value better than RobGAN.

[2] `https://github.com/xuanqing94/RobGAN`

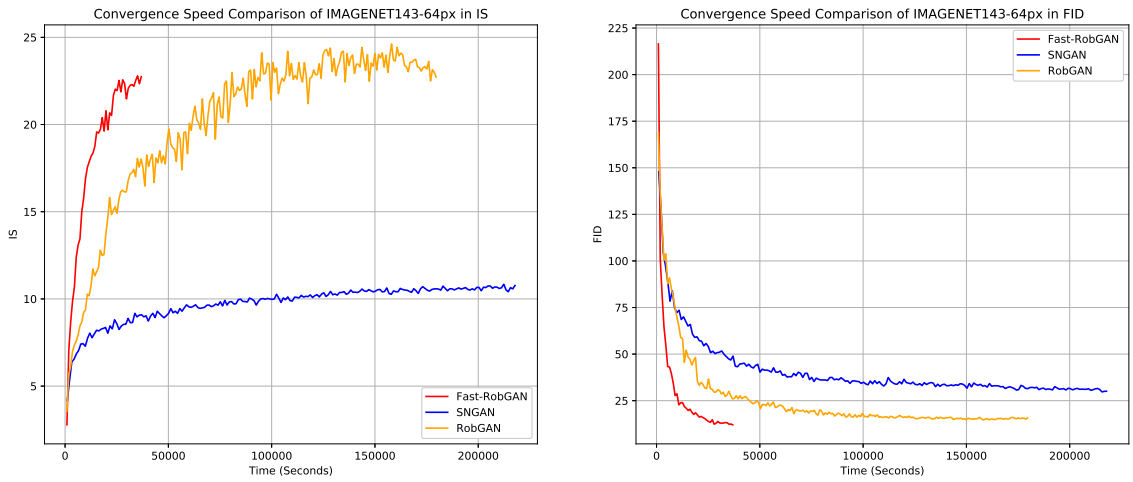

Figure 5.4: Convergence speed of IMAGENET-143 (64px)

On 128px resolution of IMAGENET-143, the results are listed in Tab. 5.4. Fast-RobGAN is able to considerably outperform SNGAN and RobGAN on both IS and FID metrics. In term of training time shown in Fig. 5.5, Fast-RobGAN greatly improves the generation quality by only spending about **20%** and **40%** overall time of SNGAN and RobGAN respectively.

On both resolutions, we also try the experiments on using the original RobGAN loss format(Eq. 2.2, 2.3) and removing the KL-divergence term. Since the α_c^g is 1.0 on the IMAGENET-143 experiments, thus, these two experiments are only different at the adversarial part in which the previous one uses the probability format and the other one uses the hinge format respectively. Overall, the experiments using the original RobGAN loss format achieves the bad FID score but with a very good IS score. To some degree, this type of intra-class model collapse is not very obvious in 64-px experiments but becomes worse in 128-px experiments. Moreover, the experiments without using the KL-divergence term also show similar results with slightly better degrees as using the original RobGAN loss format which gives the better IS score but with a worse FID score than the optimal setting. These results give the experimental supports to the effectiveness of using the hinge format adversarial part and KL-divergence term.

In term of α_c^g, though IMAGENET-143 has a relatively larger number of classes (143),

Table 5.4: Experimental results of IMAGENET-143-128px

	IS	FID	Times(sec.)
real data	53.01±0.56	0.42	N/A
SNGAN	26.12±0.29(28.2‡)	30.83	585k
RobGAN	33.81±0.47(∼30⋆)	33.98	180k
Fast-RobGAN	**40.41±0.49**	**14.48**	**71k**
Fast-RobGAN(Eq. 2.2, 2.3)	**45.94±0.52**	**25.38**	**71k**
Fast-RobGAN(w/o KL-term)	**42.54±0.40**	**17.51**	**71k**

note: ‡, ⋆ reported in 1, [LH19]. IS and FID are measured by 50k generated samples.

we are able to achieve a good result without tuning α_c^g, which is unlike the experiments in CIFAR100. We think this result is led by the natural property of the dataset. In CIFAR100, each class is very distinguishable from other classes. However, IMAGENET-143 is not randomly sub-sampled from IMAGENET and all 143 classes are different types of dogs and cats which are similar between each other. Performing conditioning is generally harder. Therefore, a larger α_c^g works for the IMAGENET-143 even the number of classes is not small.

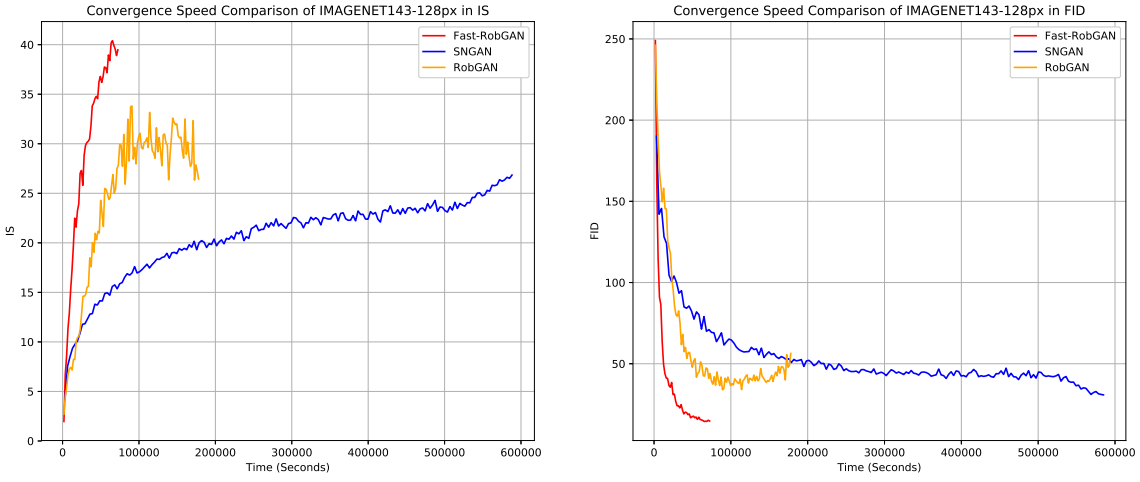

Figure 5.5: Convergence speed of IMAGENET-143 (128px)

CHAPTER 6

Conclusions

In this thesis, we study the computational limitation of using adversarial training and the generation disadvantage of the auxiliary classifier in RobGAN. Based on these problems, we propose two new improvements in order to improve the overall performance of the model:

1. A novel training algorithm which is able to efficiently perform adversarial training during the training of GAN.

2. A new improved loss objective which can effectively reduce the intra-model collapse problem in GAN framework with auxiliary classifier.

We perform experiments on 3 datasets to verify the effectiveness of our method and perform detail analysis in the results. Our new method is able to outperform previous works in all datasets in term of both generation quality and convergence speed. Moreover, we provide a new perspective to understand why adversarial training is able to help the convergence of GAN.

Due to hardware and time limitations, we are unable to perform the experiments on very large scale datasets (e.g IMAGENET with 128-px resolution) which can be seen as an important future work. Also, though our proposed method is able to achieve a good performance in experiments, the results rely on the tuning of hyperparameters which is inefficient and empirical. This leads the transferability of the settings between datasets very limited. Our proposed new method can be roughly seen as a trade-off between the conditioning and diversity of the overall system. Therefore, an automatic or dynamic balance strategy for the conditioning and diversity during the GAN training could be a good future research direction.

APPENDIX A

Supplementary Generated Samples from Models

A.1 CIFAR10

Figure A.1: Class conditional image samples for all 10 classes of CIFAR10. Each column is a class in CIFAR10. From top to down, each row is sampled from Real Data, Fast-RobGAN, SNGAN(Projection), SNGAN(AC), RobGAN

A.2 CIFAR100

Figure A.2: Class conditional image samples for all 10 classes of CIFAR100. Each column is a class in CIFAR100. From top to down, each row is sampled from Real Data, Fast-RobGAN, SNGAN(Projection), SNGAN(AC), RobGAN

REFERENCES

[ACB17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein gan." *arXiv preprint arXiv:1701.07875*, 2017.

[BDS18] Andrew Brock, Jeff Donahue, and Karen Simonyan. "Large scale gan training for high fidelity natural image synthesis." *arXiv preprint arXiv:1809.11096*, 2018.

[BLR16] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. "Neural photo editing with introspective adversarial networks." *arXiv preprint arXiv:1609.07093*, 2016.

[CZR19] Ting Chen, Xiaohua Zhai, Marvin Ritter, Mario Lucic, and Neil Houlsby. "Self-supervised gans via auxiliary rotation loss." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 12154–12163, 2019.

[DL82] DC Dowson and BV Landau. "The Fréchet distance between multivariate normal distributions." *Journal of multivariate analysis*, **12**(3):450–455, 1982.

[DSM17] Harm De Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron C Courville. "Modulating early visual processing by language." In *Advances in Neural Information Processing Systems*, pp. 6594–6604, 2017.

[GAA17] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. "Improved training of wasserstein gans." In *Advances in neural information processing systems*, pp. 5767–5777, 2017.

[GPM14] 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*, pp. 2672–2680, 2014.

[HLL18] Lei Huang, Xianglong Liu, Bo Lang, Adams Wei Yu, Yongliang Wang, and Bo Li. "Orthogonal weight normalization: Solution to optimization over multiple dependent stiefel manifolds in deep neural networks." In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[HRU17] 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*, pp. 6626–6637, 2017.

[KAL17] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. "Progressive growing of gans for improved quality, stability, and variation." *arXiv preprint arXiv:1710.10196*, 2017.

[KB14] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980*, 2014.

[Kri09] Alex Krizhevsky et al. "Learning multiple layers of features from tiny images." Technical report, Citeseer, 2009.

[KW13] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes." *arXiv preprint arXiv:1312.6114*, 2013.

[LH19] Xuanqing Liu and Cho-Jui Hsieh. "Rob-gan: Generator, discriminator, and adversarial attacker." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11234–11243, 2019.

[LTR19] Mario Lucic, Michael Tschannen, Marvin Ritter, Xiaohua Zhai, Olivier Bachem, and Sylvain Gelly. "High-fidelity image generation with fewer labels." *arXiv preprint arXiv:1903.02271*, 2019.

[LY17] Jae Hyun Lim and Jong Chul Ye. "Geometric gan." *arXiv preprint arXiv:1705.02894*, 2017.

[MK18] Takeru Miyato and Masanori Koyama. "cGANs with projection discriminator." *arXiv preprint arXiv:1802.05637*, 2018.

[MKK18] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. "Spectral normalization for generative adversarial networks." *arXiv preprint arXiv:1802.05957*, 2018.

[MLX17] 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*, pp. 2794–2802, 2017.

[MMS17] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. "Towards deep learning models resistant to adversarial attacks." *arXiv preprint arXiv:1706.06083*, 2017.

[MO14] Mehdi Mirza and Simon Osindero. "Conditional generative adversarial nets." *arXiv preprint arXiv:1411.1784*, 2014.

[NCB17] Anh Nguyen, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, and Jason Yosinski. "Plug & play generative networks: Conditional iterative generation of images in latent space." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4467–4477, 2017.

[OOS17] Augustus Odena, Christopher Olah, and Jonathon Shlens. "Conditional image synthesis with auxiliary classifier gans." In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2642–2651. JMLR. org, 2017.

[RAY16] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. "Generative adversarial text to image synthesis." *arXiv preprint arXiv:1605.05396*, 2016.

[RDS15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. "Imagenet large scale visual recognition challenge." *International journal of computer vision*, **115**(3):211–252, 2015.

[RMC15] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." *arXiv preprint arXiv:1511.06434*, 2015.

[SGZ16] 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*, pp. 2234–2242, 2016.

[SK16] Tim Salimans and Durk P Kingma. "Weight normalization: A simple reparameterization to accelerate training of deep neural networks." In *Advances in Neural Information Processing Systems*, pp. 901–909, 2016.

[SMS17] Masaki Saito, Eiichi Matsumoto, and Shunta Saito. "Temporal generative adversarial nets with singular value clipping." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2830–2839, 2017.

[SNG19] Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. "Adversarial Training for Free!" *arXiv preprint arXiv:1904.12843*, 2019.

[SVI16] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. "Rethinking the inception architecture for computer vision." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[ZGM18] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. "Self-attention generative adversarial networks." *arXiv preprint arXiv:1805.08318*, 2018.