Towards Addressing GAN Training Instabilities:

Dual-Objective GANs with Tunable Parameters

by

Kyle Otstot

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved June 2023 by the
Graduate Supervisory Committee:

Lalitha Sankar, Chair
Oliver Kosut
Giulia Pedrielli

ARIZONA STATE UNIVERSITY

August 2023

ABSTRACT

Generative Adversarial Networks (GANs) have emerged as a powerful framework for generating realistic and high-quality data. In the original "vanilla" GAN formulation, two models – the generator and discriminator – are engaged in a min-max game, where they optimize the same value function of binary cross entropy loss. Despite offering an intuitive approach, vanilla GANs often face training stability challenges such as exploding and vanishing gradients, mode collapse, and model oscillation.

Addressing these common failures, recent work has proposed the use of tunable classification losses in place of traditional value functions, given the role of the discriminator as a classifier. Although parameterized robust loss families, e.g. α-loss, have shown promising characteristics as value functions, this thesis argues that the generator and discriminator require separate objective functions to achieve their different goals. As a result, this thesis introduces the (α_D, α_G)-GAN, a parameterized class of dual-objective GANs, as an alternative approach to the standard vanilla GAN.

The (α_D, α_G)-GAN formulation, inspired by α-loss, allows practitioners to tune the parameters $(\alpha_D, \alpha_G) \in [0, \infty)^2$ to provide a more stable training process. The objectives for the generator and discriminator in (α_D, α_G)-GAN are derived, and the advantages of using these objectives are investigated. In particular, the optimization trajectory of the generator is found to be influenced by the choice of α_D and α_G.

Empirical evidence is presented through experiments conducted on various datasets, including the 2D Gaussian Mixture Ring, Celeb-A image dataset, and LSUN Classroom image dataset. Performance metrics such as mode coverage and Fréchet Inception Distance (FID) are used to evaluate the effectiveness of the (α_D, α_G)-GAN compared to the vanilla GAN and state-of-the-art Least Squares GAN (LSGAN). The experimental results demonstrate that tuning $\alpha_D < 1$ leads to improved stability, robustness to hyperparameter choice, and competitive performance compared to LSGAN.

This research contributes to advancing the field of GANs by addressing the limitations of vanilla GANs and providing theoretical insight and empirical evidence for the improved training stability of (α_D, α_G)-GAN. The findings of this thesis offer valuable insights for generating more realistic synthetic data and have the potential to drive further advancements in GAN architectures and training methodologies.

DEDICATION

In loving memory of Calli, my sweet golden retriever, whose playful spirit and unwavering love brought warmth and happiness to everyone around her.

ACKNOWLEDGMENTS

I would like to acknowledge the Sankar Lab of Arizona State University for their valuable contributions to this thesis. This research builds upon their recent work, and I am grateful for their efforts. Special thanks go to Dr. Lalitha Sankar, Monica Welfert, and Dr. Gowtham Kurri for their guidance, support, and collaboration throughout the project. I am also grateful to the National Science Foundation (NSF) for their financial support, provided through grants CIF-1901243, CIF-1815361, CIF-2007688, CIF-2134256, SaTC-2031799, and SCH-2205080. These grants have played a crucial role in funding the lab's activities and enabling the research presented in this thesis.

I would like to acknowledge the insightful discussions and feedback from my adviser, colleagues, and friends, which have greatly influenced the development of this work. Lastly, I would like to express my appreciation to my family for their support and belief in me throughout my academic journey. I extend my sincere thanks to all those mentioned above and anyone else who has contributed in any way.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Generative Adversarial Networks (GANs) have emerged as a powerful framework for generating realistic and high-quality data across a variety of applications, including art generation (Gatys, Ecker, and Bethge 2015), audio synthesis (Donahue, McAuley, and Puckette 2018), and healthcare (Jiang et al. 2020). The concept of GANs was first introduced by (Goodfellow et al. 2014), and since then, GANs have gained significant attention in the field of deep learning. The original GAN, known as *vanilla GAN*, consists of a generator network and discriminator network: the generator learns a mapping from random noise to synthetic data samples, while the discriminator estimates the probabilities of samples being real or generated (refer to Figure 1). The training objective of vanilla GAN involves a min-max game between the generator and the discriminator; specifically, the generator aims to produce samples that resemble the real data, while the discriminator aims to classify the source of each sample.

Although the objectives of vanilla GANs provide an intuitive formulation for training, they can lead to several challenges. One common challenge is the threat of exploding and vanishing gradients: imbalanced performances in GAN training often coincide with the occurrence of these gradient-related issues. For example, gradient explosion occurs when the generator produces severely misclassified samples, leading to unstable updates and overshooting of the generated data. On the other hand, gradient vanishing occurs when discriminator confidently assigns near-zero probabilities to the generator, hindering meaningful weight updates due to the saturating nature of the generator objective. Another challenge faced by the vanilla GAN is mode collapse,

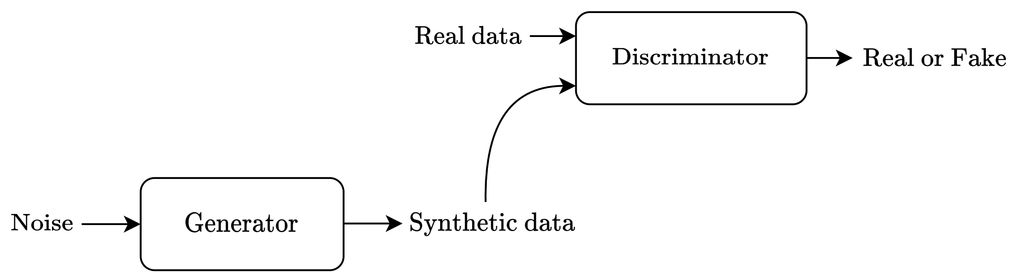


Figure 1. A simple depiction of a Generative Adversarial Network (GAN). The generator model aims to produce synthetic samples that trick the discriminator model into classifying them as real.

where the generator produces samples that resemble only a small portion of the real data. In this scenario, the generator lacks the incentive to capture the full diversity of the data, as the discriminator exhibits a locally flat decision landscape. Additionally, model oscillation may occur when the steep gradients of outlier samples heavily influence the generator weight update, at the expense of preserving the proximity of the samples close to the real data; this leads to an iterative oscillation of the generator and discriminator without any significant model improvement.

While alternative GAN architectures have been proposed to address these instabilities (Bhatia et al. 2021; Nowozin, Cseke, and Tomioka 2016)they often have limited success in fully overcoming the challenges. As a result, this thesis aims to explore an alternative approach to improve the stability of GAN training and the quality of generated samples. Specifically, we introduce the (α_D, α_G)-GAN as a descendant of α-loss (Sypherd et al. 2019) and generalization of α-GAN (Kurri, Sypherd, and Sankar 2021; Kurri et al. 2022). The (α_D, α_G)-GAN formulation aims to overcome the limitations of the vanilla GAN by allowing the practitioner to tune parameters α_D and α_G for a more stable training process. We derive specific objectives for the generator and discriminator in the (α_D, α_G)-GAN and investigate the advantages of using these objectives; in doing so, we compare (α_D, α_G)-GAN to the vanilla GAN with gradient analysis, and find that tuning α_D and α_G has an impact on the optimization trajectory traversed by the generator.

Furthermore, we present empirical evidence from experiments conducted on various datasets, including a 2D Gaussian Mixture Ring (Srivastava et al. 2017), the Celeb-A image dataset (Liu et al. 2015), and the LSUN Classroom image dataset (Yu et al. 2015). Multiple evaluation metrics, including mode coverage and Fréchet Inception Distance (Heusel et al. 2017), are used to measure the performance of (α_D, α_G)-GAN compared to the vanilla GAN and state-of-the-art Least Squares GAN (LSGAN) (Mao et al. 2017). Experiments are conducted for numerous seeds on a range of learning rates and training epochs, which are particularly sensitive to GAN training instabilities. The experimental results demonstrate the effectiveness and stability of the (α_D, α_G)-GAN architecture in generating diverse and high-quality samples. In particular, tuning $\alpha_D < 1$ exhibits robustness to hyperparameter choice and achieves competitive performance compared to LSGAN.

Overall, this thesis aims to address the common failures of vanilla GANs by introducing the (α_D, α_G)-GAN architecture. By formulating dual objectives for the generator and discriminator, and conducting extensive experiments, we provide theoretical insight and empirical evidence for the improved training stability of (α_D, α_G)-GAN. We hope the findings of this research contribute to advancing the field of GANs and offer valuable insights for generating more realistic synthetic data.

Chapter 2

THE FAILURES OF VANILLA GAN

Generative Adversarial Networks (GANs) have shown great potential in generating realistic and diverse samples. However, the original formulation of GANs, known as *vanilla GAN* (Goodfellow et al. 2014), is susceptible to several common failures that hinder their training and affect the quality of generated samples. In this section, we first define the vanilla GAN objectives for both the discriminator and generator, then discuss the specific failures of vanilla GAN: these include exploding and vanishing gradients, mode collapse, and model oscillation.

2.1 Vanilla GAN Objectives

As depicted in Figure 2, the vanilla GAN framework consists of two modules: a generator network G parameterized by vectors $\theta \in \Theta \subset \mathbb{R}^{n_g}$, and a discriminator network D parameterized by $\omega \in \Omega \subset \mathbb{R}^{n_d}$. The generator $G_\theta : \mathcal{Z} \to \mathcal{X}$ maps noise $Z \sim P_Z$ to a data sample in $\mathcal{X}$ and aims to produce synthetic samples that resemble real data. The discriminator $D_\omega : \mathcal{X} \to [0, 1]$ estimates the probability that input in $\mathcal{X}$ is drawn from P_r (real) as opposed to P_{G_θ} (generated), thus serving to distinguish between real and generated samples. Originally, the GAN training objective involved a zero-sum, min-max game between the two networks, where the generator and discriminator optimize their objectives adversarially. The generator's objective is to minimize the discriminator's ability to correctly classify between the real and generated samples, while the discriminator's objective is to maximize its classification accuracy,

$$\inf_{\theta\in\Theta}\sup_{\omega\in\Omega} V(\theta,\omega), \tag{2.1}$$

where $V(\theta,\omega)$ is a value function measuring the classification accuracy of discriminator D_ω between the real samples, and samples produced by generator G_θ. For the vanilla GAN, (Goodfellow et al. 2014) use the negative *binary cross entropy* (BCE) loss as the value function

$$V_{\text{VG}}(\theta,\omega) = \mathbb{E}_{X\sim P_r}[\log D_\omega(X)] + \mathbb{E}_{X\sim P_{G_\theta}}[\log(1-D_\omega(X))]. \tag{2.2}$$

This min-max game creates a competitive dynamic between the generator and discriminator, driving them to improve iteratively. However, while the vanilla GAN objectives offer an intuitive formulation for training, they can also contribute to challenges such as exploding and vanishing gradients, mode collapse, and model oscillation, which ultimately need to be addressed to achieve more stable and effective GAN training.

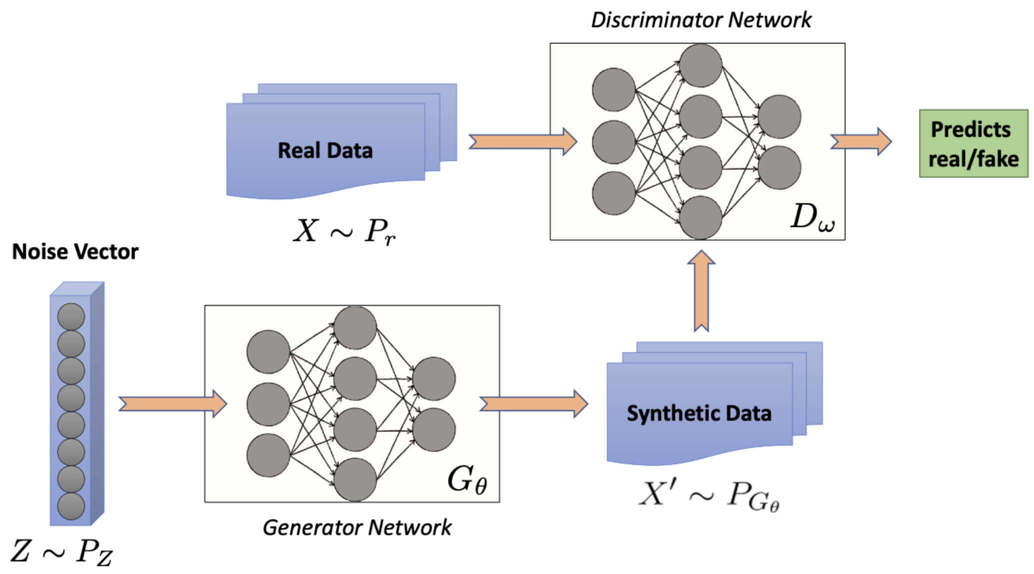


Figure 2. The GAN architecture consists of a generator network G_θ and discriminator network D_ω engaged in an adversarial game.

2.2 Exploding & Vanishing Gradients

During vanilla GAN training, imbalanced performances between the generator and discriminator often coincide with the presence of exploding and vanishing gradients. When updating the generator weights during the backward pass of the network $D_\omega \circ G_\theta$, the gradients are computed by propagating the error signal from the output layer of D_ω to the input layer of G_θ, following the chain rule of derivatives. Each layer contributes to the gradient update by multiplying the incoming gradient with the local gradient of its activation function, and passing it to the preceding layer.

When the gradients become large, the successive multiplication of these gradients across the layers can result in an exponential growth, known as gradient explosion. Conversely, small gradients can lead to an exponential decay, referred to as gradient vanishing. In both cases, networks with multiple hidden layers are particularly susceptible to unstable weight updates, causing extremely large or small values that may overflow or underflow the numerical range of computations, respectively.

In the context of vanilla GANs, gradient explosion can occur when the generator successfully produces samples that are severely misclassified (close to 1) by the discriminator. During training, the generator is updated using the loss function $\log(1 - D_\omega(x))$, which diverges to $-\infty$ as the discriminator output $D_\omega(x)$ approaches 1. Consequently, the gradients for the generator weights fail to converge to non-zero values, leading to the generated data potentially overshooting the real data in any direction. In severe cases of gradient explosion, the weight update can direct the generated data toward a region far from the real data. As a result, the discriminator can easily assign zero probabilities to the generated data and ones to the real data. As the discriminator output approaches zero, the generator's loss function converges to zero, causing the gradients of the generator weights to vanish gradually. As shown in Figure 3(a), this phenomenon can prevent the generator from effectively correcting itself and improving its performance over time.

Addressing the issues of exploding and vanishing gradients, (Goodfellow et al. 2014) proposed a solution in the form of a *non-saturating* (NS) generator objective:

$$V_{\mathrm{VG}}^{\mathrm{NS}}(\theta, \omega) = \mathbb{E}_{X \sim P_{G_\theta}}[-\log D_\omega(X)]. \tag{2.3}$$

The use of this non-saturating loss function allows the generated data to converge toward the real distribution. As the discriminator output approaches 1, the generator loss approaches zero, indicating that the generated data becomes more aligned with the real distribution. Additionally, with a high-performing discriminator, the generator receives steep gradients (as opposed to vanishing gradients) during the update process; this occurs because the generator loss diverges to $+\infty$ as the discriminator output approaches zero.

Although the non-saturating vanilla GAN (an industry standard) incorporates different objective functions for the generator and discriminator, it still suffers from mode collapse and oscillations (Arjovsky and Bottou 2017; Wiatrak, Albrecht, and Nystrom 2019) which is discussed in the next section. These issues arise due to convergence problems and the sensitivity of the GAN to hyperparameter initialization, resulting from the presence of large gradients. Various alternative dual-objective GANs have been proposed, such as the Least Squares GAN (LSGAN) (Mao et al. 2017), RényiGAN (Bhatia et al. 2021), non-saturating f-GAN (Nowozin, Cseke, and Tomioka 2016), and hybrid f-GAN (**poole2016improved**). However, these approaches have rarely been successful in fully addressing GAN training instabilities.

2.3 Mode Collapse & Model Oscillation

During GAN training, the primary goal of the generator is to produce high-quality samples that encompass the full range of diversity found in the real distribution. However, there is a potential drawback known as *mode collapse*, which occurs when the generator produces samples that closely resemble only a limited subset of the real data. In such cases, the generator lacks the incentive to capture the remaining modes since the discriminator struggles to effectively differentiate between the real and generated samples. One possible explanation for this phenomenon, as depicted in Figure 3(b), is that the generator and/or discriminator become trapped in a local minimum, impeding the necessary adjustments to mitigate mode collapse. In the figure, the cluster of generated data approaches a single mode in the real distribution, which forces the discriminator to adjust properly; if the discriminator landscape is sufficiently flat in the mode neighborhood, then the generator may struggle to adapt.

In the case of mode collapse, the generator or discriminator may converge prematurely, leading to a suboptimal standstill between the models. On the other hand, a generator training with the non-saturating value function $V_{\mathrm{VG}}^{\mathrm{NS}}$ may experience a complete failure to converge due to influence from outlier generated data. Illustrated in Figure 3(c), most of the generated data occupies the real modes and could receive small gradients $\partial \ell_{\mathrm{BCE}}\left(1, D_\omega(x)\right) / \partial x$ if the discriminator landscape is locally flat. However, some outlier data is situated very far from the real distribution and consequently receive steep gradients. To address the high non-saturating loss from the outliers, the generator prioritizes directing the outlier data toward the real data over keeping the close data in place; as a result, the generator update reflects a compromise in

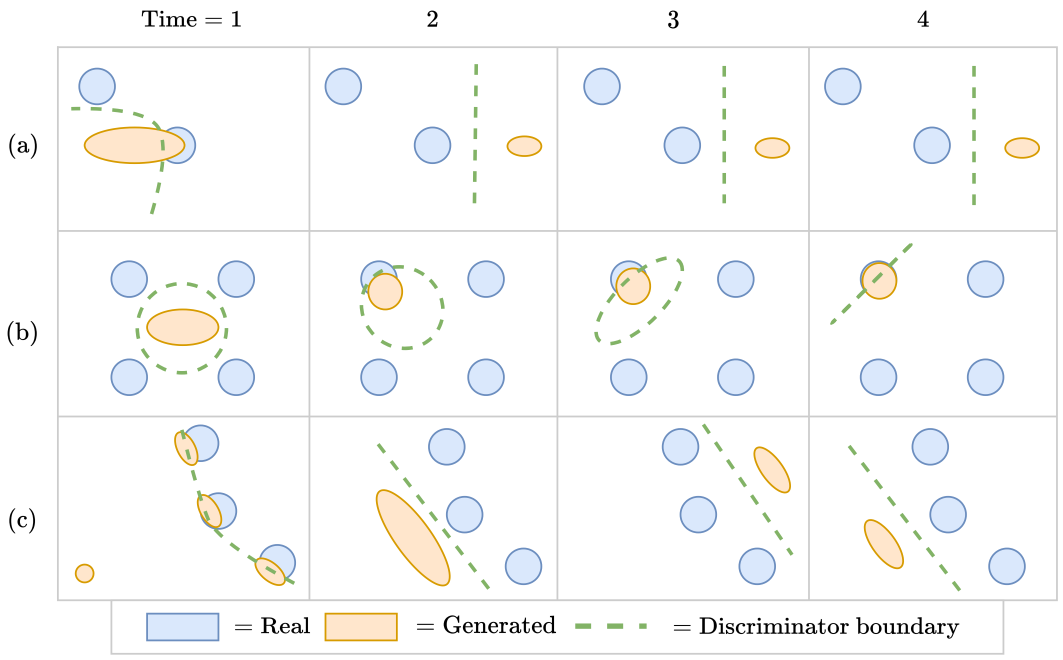


Figure 3. An illustration of three common GAN failures – (a) exploding and vanishing gradients, (b) mode collapse, and (c) model oscillation – over 4 points in training time.

Time = 2 of Figure 3(c), where the outliers are resolved at the expense of nudging the other data away from the modes.

Although the generator succeeds at bringing down the average loss by eliminating these outliers, the discriminator is now able to confidently distinguish between the distributions, leading to near-zero probabilities assigned to the generated data. In turn, the generated samples all receive steep gradients which may result in oscillations around the real data. Gradually, the models lose their accumulated knowledge on the structure of the real distribution and essentially restart the training process.

Chapter 3

FORMULATING THE (α_D, α_G)-GAN

In the previous section, we present an overview of the numerous challenges faced by the vanilla GAN when training with the value function in (2.2). Addressing these challenges, we introduce a generalization of the vanilla GAN, namely (α_D, α_G)-GAN, that employs a pair of tunable objectives, each possessing characteristics that help enhance the stability of the training process. First, we derive (α_D, α_G)-GAN and define its objectives for the generator and discriminator; next, we explore the intuitive advantages of using these novel objectives to help improve GAN training stability.

3.1 (α_D, α_G)-GAN: A Dual-Objective Generalization of α-GAN

Similar to the vanilla GAN, the (α_D, α_G)-GAN consists of a generator network and a discriminator network engaged in an adversarial game. As shown in (2.1), the adversarial game can be formulated as a zero-sum min-max problem for a chosen value function $V(\theta, \omega)$. Observing that the discriminator is a classifier, (Kurri, Sypherd, and Sankar 2021) recently demonstrated that the value function can be expressed using a class probability estimation (CPE) loss denoted as $\ell(y, \hat{y})$, where $y \in \{0, 1\}$ represents the true label and $\hat{y} \in [0, 1]$ denotes the soft prediction of y, given below:

$$V(\theta, \omega) = \mathbb{E}_{X \sim P_r}[-\ell(1, D_\omega(X))] + \mathbb{E}_{X \sim P_{G_\theta}}[-\ell(0, D_\omega(X))]. \tag{3.1}$$

Using this approach, the authors introduce α-GAN using the tunable CPE loss α-loss (Sypherd et al. 2019; Sypherd et al. 2022), defined for $\alpha \in (0, \infty]$ as

$$\ell_\alpha(y, \hat{y}) := \frac{\alpha}{\alpha - 1}\left(1 - y\hat{y}^{\frac{\alpha-1}{\alpha}} - (1-y)(1-\hat{y})^{\frac{\alpha-1}{\alpha}}\right). \tag{3.2}$$

The findings demonstrate that the α-GAN formulation can recover various f-divergence based GANs, such as the Hellinger GAN (Nowozin, Cseke, and Tomioka 2016) with $\alpha = 1/2$, the vanilla GAN (Goodfellow et al. 2014) with $\alpha = 1$, and the Total Variation (TV) GAN (Nowozin, Cseke, and Tomioka 2016) with $\alpha = \infty$. Additionally, when considering a sufficiently large discriminator set Ω, the min-max optimization for α-GAN in (2.1) simplifies to minimizing the Arimoto divergence between the real and generated distributions P_r, P_{G_θ} (Österreicher 1996; Liese and Vajda 2006).

Despite the advantages offered by each of the aforementioned GANs, they still suffer from various types of training instabilities outlined in Chapter 2. Recent research has argued that α-loss exhibits favorable gradient behaviors for different α values (Sypherd et al. 2022). It also ensures the development of robust classifiers that can reduce the confidence of the discriminator (a classifier), thereby enabling the generator to learn without gradient issues. To this end, we propose the use of distinct α-loss objectives for each player; specifically, we introduce a tunable dual-objective (α_D, α_G)-GAN, where the objective functions of the discriminator and generator are formulated using α-loss with parameters $\alpha_D \in (0, \infty]$ and $\alpha_G \in (0, \infty]$, respectively. In particular, the discriminator maximizes $V_{\alpha_D}(\theta, \omega)$ while the generator minimizes $V_{\alpha_G}(\theta, \omega)$, where

$$V_\alpha(\theta, \omega) = \mathbb{E}_{X \sim P_r}[-\ell_\alpha(1, D_\omega(X))] + \mathbb{E}_{X \sim P_{G_\theta}}[-\ell_\alpha(0, D_\omega(X))]. \tag{3.3}$$

Notice that we recover the α-GAN (Kurri, Sypherd, and Sankar 2021; Kurri et al. 2022) value function when $\alpha_D = \alpha_G$ and the vanilla GAN value function when $\alpha_D = \alpha_G = 1$.

The resulting (α_D, α_G)-GAN is given by

$$\sup_{\omega\in\Omega} V_{\alpha_D}(\theta,\omega) \tag{3.4a}$$

$$\inf_{\theta\in\Theta} V_{\alpha_G}(\theta,\omega). \tag{3.4b}$$

Considering that α-GAN recovers various well-known GANs – including the vanilla GAN, which is prone to saturation – the (α_D, α_G)-GAN formulation using the generator objective in (3.3) can similarly saturate early in training, potentially leading to vanishing gradients. Thus, we propose the following *non-saturating* alternative to the generator's objective in (3.3):

$$V^{\text{NS}}_{\alpha_G}(\theta,\omega) = \mathbb{E}_{X\sim P_{G_\theta}}[\ell_{\alpha_G}(1, D_\omega(X))], \tag{3.5}$$

thereby replacing (3.4b) with

$$\inf_{\theta\in\Theta} V^{\text{NS}}_{\alpha_G}(\theta,\omega). \tag{3.6}$$

Henceforth, we will refer to the (α_D, α_G)-GANs training with the generator objectives V_{α_G} (3.3) and $V^{\text{NS}}_{\alpha_G}$ (3.5) as the saturating (α_D, α_G)-GAN and non-saturating (α_D, α_G)-GAN, respectively.

3.2 Analysis of (α_D, α_G)-GAN

For both saturating and non-saturating (α_D, α_G)-GAN, the discriminator seeks to maximize the value function $V_{\alpha_D}(\theta,\omega)$ with respect to its parameters $\omega \in \Omega$. Assuming a sufficiently large discriminator set Ω, (Kurri, Sypherd, and Sankar 2021) show that for a fixed generator G_θ, the discriminator optimizing the sup of $V_{\alpha_D}(\theta,\omega)$ is given by

$$D_{\omega^*}(x) = \frac{p_r(x)^{\alpha_D}}{p_r(x)^{\alpha_D} + p_{G_\theta}(x)^{\alpha_D}}, \tag{3.7}$$

where p_r and p_{G_θ} are the corresponding densities of the distributions P_r and P_{G_θ}, respectively, with respect to a base measure dx (e.g., Lebesgue measure).

We note that the input to the discriminator is a random variable X which can be viewed as being sampled from a mixture distribution, i.e., $X \sim \delta P_r + (1-\delta)P_{G_\theta}$ where $\delta \in (0,1)$. Without loss of generality, we assume $\delta = 1/2$ but the analysis that follows can be generalized for arbitrary δ. We use the Bernoulli random variable $Y \in \{0,1\}$ to indicate that $X = x$ is from the real $Y = 1$ or generated $Y = 0$ distributions. Therefore, $P(Y=1) = 1 - P(Y=0) = \delta = 1/2$. Thus, one can then compute the posterior $P(Y=1|X=x)$ as follows:

$$P(Y=1|X=x) = \frac{\delta P_r(x)}{\delta P_r(x) + (1-\delta)P_{G_\theta}(x)} \tag{3.8}$$

$$\overset{\delta=1/2}{=} \frac{P_r(x)}{P_r(x) + P_{G_\theta}(x)} \tag{3.9}$$

where the simplification in (3.9) follows from the fact that we assume $\delta = 1/2$. In the following theorem, we observe that the optimal discriminator D_{ω^*} for the (α_D, α_G)-GAN in (3.7) is simply the α_D-tilted[1] version of $P(Y=1|X=x)$.

Theorem 3.2.1. *Assuming a sufficiently large discriminator set Ω, the optimal (α_D, α_G)-GAN discriminator $D_{\omega^*}(x)$ defined in* (3.7) *is equivalent to the α_D-tilted version of the true posterior $P(Y=1|X)$, namely $P_{\alpha_D}(Y=1|X)$.*

Proof sketch. We multiply the numerator and denominator of $D_{\omega^*}(x)$ in (3.7) by $1/\left(p_r(x) + p_{G_\theta}(x)\right)^{\alpha_D}$, then simplify and substitute via (3.9), which leads to $P_{\alpha_D}(Y = 1|X)$ with more simplification. See Appendix A.1 for a detailed proof.

Choosing $\alpha_D = 1$ clearly recovers the true posterior and in theory, should suffice to learn the optimal discriminator (provided Ω is sufficiently large). However, in practice,

[1]In information theory and statistics, the term α-tilting refers to raising the entries of a probability vector to their α^{th}-power and normalizing them to obtain a probability measure.

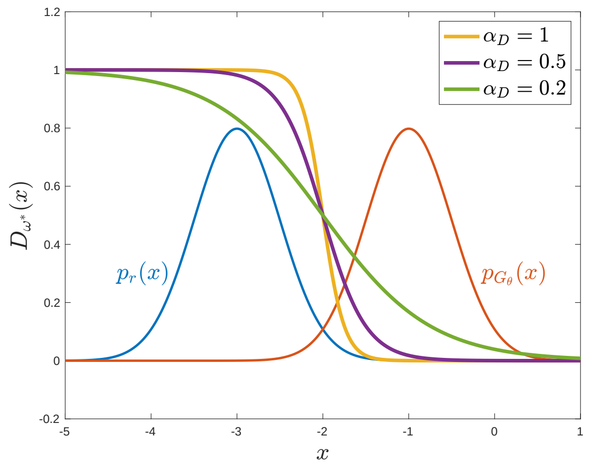


Figure 4. A toy example with a real distribution $P_r = \mathcal{N}(-2, 0.5^2)$ (blue) and generated distribution $P_{G_\theta} = \mathcal{N}(2, 0.5^2)$ (orange).The optimal discriminator output $D_{\omega^*}(x)$ in (3.7) is plotted for several values of $\alpha_D \leq 1$.

tuning α_D allows us address the training challenges discussed earlier. We first begin by understanding several important consequences of tilting the true posterior. As α_D is tuned above 1, (3.7) implies a more confident discriminator, i.e. if $p_{G_\theta}(x) > p_r(x)$, then D_{ω^*} decays more rapidly from 1/2, and if $p_{G_\theta}(x) < p_r(x)$, then D_{ω^*} increases more rapidly from 1/2. In the extreme case of $\alpha_D \to \infty$, the discriminator output for input x simplifies to $\mathbb{1}\{p_r(x) > p_{G_\theta}(x)\} + \frac{1}{2}\mathbb{1}\{p_r(x) = p_{G_\theta}(x)\}$, thus implementing the Maximum Likelihood decision rule. Conversely, tuning $\alpha_D < 1$ induces a cautious discriminator with more uniform predictions ($D_{\omega^*}(x) \to 1/2$). In the extreme case of $\alpha_D = 0$, the discriminator output is 1/2 for all input.

Figure 4 illustrates the difference between three α_D-tilted posteriors with $\alpha_D \leq 1$ for a toy example consisting of two Gaussian densities. As α_D is tuned below 1 in the value function $V_{\alpha_D}(\theta, \omega)$, the discriminator $D_{\omega^*}(x)$ becomes less confident in its predictions, leading to "smoother" and less discrete output. In the $\alpha_D = 1$ case, where the discriminator trains with binary cross entropy loss, we observe a very flat discriminator output in the region densely populated by p_{G_θ}; this poses a problem for generated data x that deviates significantly from the real data, since the gradient $\partial\left[D_{\omega^*}(x)/\,\partial x\right]$ is close to zero. Consequently, the generator gradient with respect to its weight vector θ

$$-\frac{\partial \ell_{\alpha_D}\left(0, D_{\omega^*}(x)\right)}{\partial \theta} = -\frac{\partial \ell_{\alpha_D}\left(0, D_{\omega^*(x)}\right)}{\partial D_{\omega^*}(x)} \times \frac{\partial D_{\omega^*}(x)}{\partial x} \times \frac{\partial x}{\partial \theta} \tag{3.10}$$

also tends to zero, resulting in minimal or no weight updates. When dealing with high-dimensional data, oftentimes the discriminator can easily distinguish between real and generated data; this leads to very confident, discrete predictions when trained with $\alpha_D \geq 1$. As a result, the generator may receive limited or insufficient feedback from the discriminator, resulting in a hindered ability to learn the real distribution. Intuitively, tuning α_D below 1 restricts the confidence of the discriminator while retaining necessary information about the data. By slowing down the learning of the discriminator, the generator receives informative gradients throughout the training process, allowing it to keep pace and improve iteratively alongside the discriminator.

3.2.1 The Saturating (α_D, α_G)-GAN

The generator for saturating (α_D, α_G)-GAN seeks to minimize the value function $V_{\alpha_G}(\theta, \omega)$ in (3.3), while the generator for non-saturating (α_D, α_G)-GAN minimizes $V_{\alpha_G}^{NS}(\theta, \omega)$ in (3.5). In both cases, the optimization trajectory traversed by the generator during training is strongly dependent on the practitioner's choice of $(\alpha_D, \alpha_G) \in [0, \infty)^2$. In the following two theorems, we offer deeper insights into how the optimization trajectory is influenced by tuning the saturating and non-saturating (α_D, α_G)-GANs, respectively; at the level of a single generated sample x with the corresponding random variable $X \sim P_{G_\theta}$, we find that the (α_D, α_G) parameters have

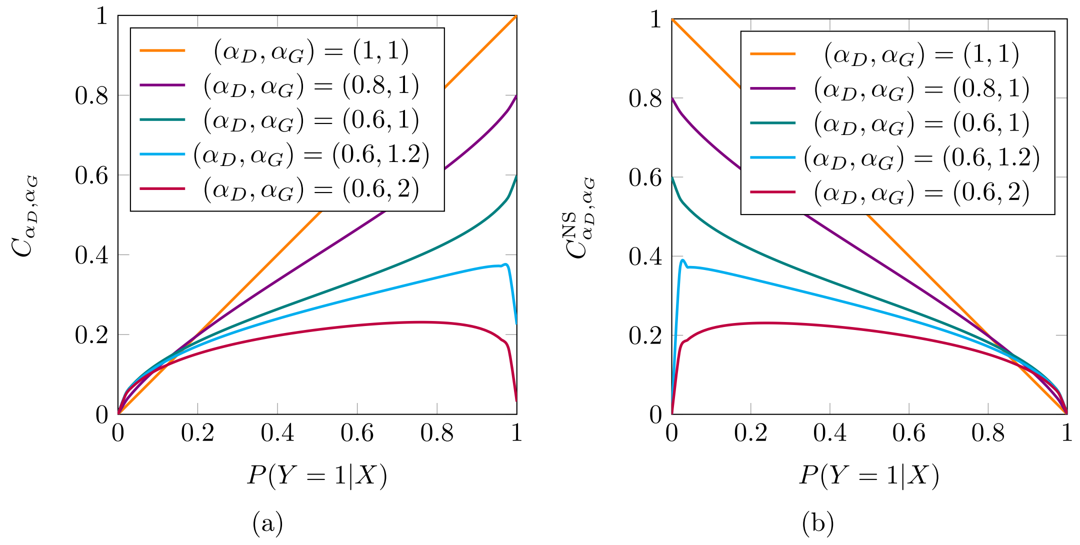


Figure 5. (a) Plot of the gradient scalar C_{α_D,α_G} defined in (3.12) over the true posterior $P(Y = 1|X)$ for five different saturating (α_D, α_G)-GANs. (b) Plot of the gradient scalar $C^{\text{NS}}_{\alpha_D,\alpha_G}$ defined in (3.14) over the true posterior $P(Y = 1|X)$ for five different non-saturating (α_D, α_G)-GANs.

no effect on the direction of the (i) saturating gradient $-\partial\ell_{\alpha_G}(0, D_{\omega^*}(x))/\partial x$ and (ii) non-saturating gradient $\partial\ell_{\alpha_G}(1, D_{\omega^*}(x))/\partial x$, but they do have a significant effect on the gradient magnitudes.

Theorem 3.2.2. *Let x be a sample generated by G_θ, and D_{ω^*} be optimal with respect to $V_{\alpha_D}(\theta, \omega)$. Then the direction of the* ***saturating*** *gradient $-\partial\ell_{\alpha_G}(0, D_{\omega^*}(x))/\partial x$ is independent of α_D and α_G.*

Proof sketch. We first substitute the optimal discriminator $D_{\omega^*}(x)$ expression defined in (3.7) into the generator objective $-\ell_{\alpha_G}(0, D_\omega(x))$, then take the derivative with respect to x. We simplify the expression with the quotient rule and ultimately arrive at the following equation:

$$-\frac{\partial\ell_{\alpha_G}(0, D_{\omega^*}(x))}{\partial x} = C_{\alpha_D,\alpha_G}\left(\frac{1}{p_{G_\theta}(x)}\frac{\partial p_{G_\theta}}{\partial x} - \frac{1}{p_r(x)}\frac{\partial p_r}{\partial x}\right), \tag{3.11}$$

where C_{α_D,α_G} is a scalar defined as

$$C_{\alpha_D,\alpha_G} = \alpha_D P_{\alpha_D}\left(Y=1|X=x\right)\left(1-P_{\alpha_D}\left(Y=1|X=x\right)\right)^{1-1/\alpha_G}. \tag{3.12}$$

A detailed proof can be found in Appendix A.2.

The gradient of the generator loss with respect to a generated sample can be interpreted as a vector suggesting which direction (and magnitude) the generator should nudge the sample in order to reduce the loss. Theorem 3.2.2 demonstrates that tuning the saturating (α_D, α_G)-GAN has no impact on the direction of the nudge, but it does have influence on the magnitude, which is proportional to C_{α_D,α_G}. In Figure 5(a), we plot this scalar over the true probability that $X \sim \frac{1}{2}P_r + \frac{1}{2}P_{G_\theta}$ is real, namely $P(Y=1|X)$, for five notable (α_D, α_G) combinations. In the $(1,1)$ case (i.e., vanilla GAN), the gradient scalar approaches zero for samples far away from the real data, and linearly increases to 1 for samples closer to the real data. As discussed earlier, this optimization strategy is troublesome for early periods of training when the real and generated data are fully separable, since the sample gradients are essentially zeroed out by the scalar. To address this issue, Figure 5(a) shows that we can tune α_D below 1 (e.g., 0.6) to ensure that samples most likely to be "generated" $(P(Y=1|X) \to 0)$ receive sufficient gradient for updates that direct them closer to the real distribution.

Moreover, the saturating $(1,1)$-GAN suffers with convergence issues since generated samples close to the real data receive gradients large in magnitude $(C_{1,1} \to 1)$. Ideally, these generated samples should not be instructed to move since they convincingly pass as real to the optimal discriminator. As explained in Section 2.2, an excessive gradient can nudge the generated data away from the real data, which ultimately separates the distributions and forces the GAN to restart training. Although the $(0.6,1)$-GAN in Figure 5(a) appears to restrain the gradient scalar of samples close to the real data ($P(Y=1|X) \to 1$), we observe that tuning α_G above 1 allows this gradient to converge to zero as desired. This trend is exemplified in Figure 6(a), where GANs with $\alpha_D < 1$ and $\alpha_G > 1$ train with a generator loss that (i) possesses a meaningful gradient in the dense p_{G_θ} region, and (ii) is convex in the dense p_r region.

3.2.2 The Non-Saturating (α_D, α_G)-GAN

Although tuning the saturating (α_D, α_G)-GAN away from vanilla GAN promotes a more favorable optimization trajectory for the generator, the loss $-\ell_{\alpha_G}(0, D_\omega(x))$ altogether seems counter-intuitive; realistically, generated samples far from P_r should receive larger gradients than samples close to P_r. (Goodfellow et al. 2014) acknowledge this limitation and propose the non-saturating value function $V_{\mathrm{VG}}^{\mathrm{NS}}$ in (2.3), which we generalize to $V_{\alpha_G}^{\mathrm{NS}}$ in (3.3). The following theorem considers the loss function $\ell_{\alpha_G}(1, D_\omega(x))$ used in $V_{\alpha_G}^{\mathrm{NS}}$, and finds that its gradient with respect to x is not influenced by our choice of (α_D, α_G). We discuss the implications in the sequel.

Theorem 3.2.3. *Let x be a sample generated by G_θ and D_{ω^*} be optimal with respect to $V_{\alpha_D}(\theta, \omega)$. Then the direction of the* ***non-saturating*** *gradient $\partial \ell_{\alpha_G}(1, D_{\omega^*}(x)) / \partial x$ is independent of α_D and α_G.*

Proof sketch. Similar to Theorem 3.2.2, we substitute the optimal discriminator $D_{\omega^*}(x)$ expression defined in (3.7) into $\ell_{\alpha_G}(1, D_\omega(x))$, then take the derivative with respect to x. We simplify via the quotient rule and arrive at the following equation:

$$\frac{\partial \ell_{\alpha_G}(1, D_{\omega^*}(x))}{\partial x} = C_{\alpha_D,\alpha_G}^{\mathrm{NS}} \left(\frac{1}{p_{G_\theta}(x)} \frac{\partial p_{G_\theta}}{\partial x} - \frac{1}{p_r(x)} \frac{\partial p_r}{\partial x} \right), \tag{3.13}$$

where $C_{\alpha_D,\alpha_G}^{\mathrm{NS}}$ is a scalar defined as

$$C_{\alpha_D,\alpha_G}^{\mathrm{NS}} = \alpha_D \left(1 - P_{\alpha_D}(Y=1|X=x)\right) P_{\alpha_D}(Y=1|X=x)^{1-1/\alpha_G}. \tag{3.14}$$

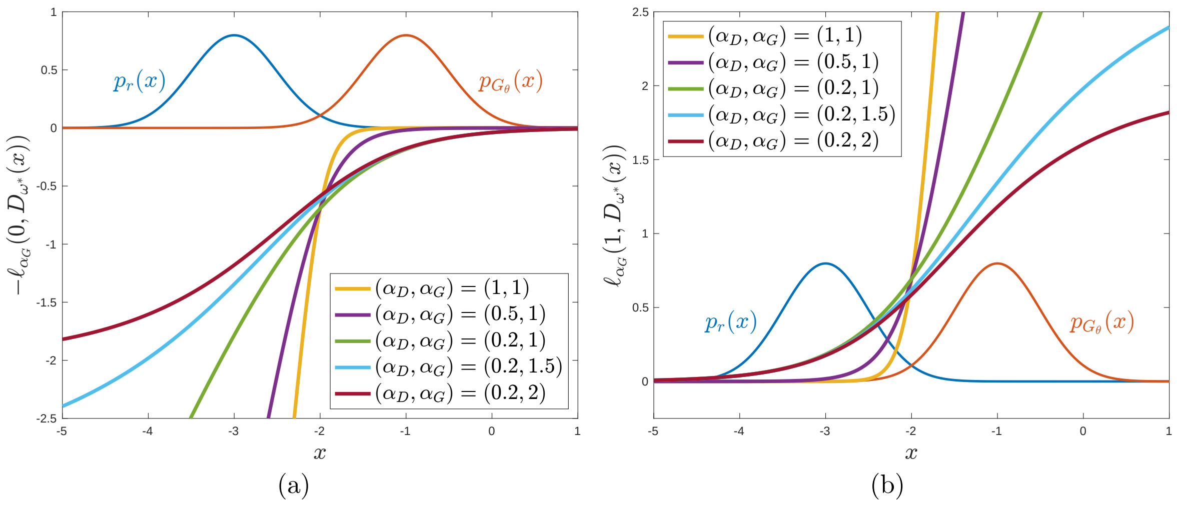


Figure 6. (a) Plot of the saturating generator loss $-\ell_{\alpha_G}(0, D_{\omega^*}(x))$ for several values of $\alpha_D \leq 1, \alpha_G \geq 1$. (b) Plot of the non-saturating generator loss $\ell_{\alpha_G}(1, D_{\omega^*}(x))$ for several values of $\alpha_D \leq 1, \alpha_G \geq 1$.

A detailed proof can be found in Appendix A.3.

The non-saturating (α_D, α_G)-GAN deviates from the vanilla GAN by allowing the practitioner to tune α_D and α_G in order to stabilize the optimization trajectory traversed by the generator. In Theorem 3.2.3, we show that tuning these parameters exclusively influence the magnitude of the gradients ($\propto C^{\mathrm{NS}}_{\alpha_D,\alpha_G}$) received by the generated samples during backpropagation. Building on this, Figure 5(b) illustrates the relationship between the gradient scalar $C^{\mathrm{NS}}_{\alpha_D,\alpha_G}$ and the probability that a sample $X \sim \frac{1}{2}P_r + \frac{1}{2}P_{G_\theta}$ is real, namely $P(Y = 1|X)$, for several select instances of (α_D, α_G). In the vanilla $(1, 1)$-GAN case, we observe a negative linear relationship: the samples least likely to be real ($P(Y = 1|X) \to 0$) receive steep gradients ($C^{\mathrm{NS}}_{1,1} \to 1$) while the samples most likely to be real receive minimal gradients. Although intuitive, the vanilla GAN's optimization strategy may render it vulnerable to model oscillation, a common GAN failure detailed in section 2.3. In this scenario, the steep gradients of the outlier (far from real) samples heavily influence the generator weight update, which can potentially draw the other generated samples away from the real modes. If successful, the generator reduces the average non-saturating loss at the expense of the data's separability; in the next iteration, the discriminator confidently assigns near-zero probabilities to the generated samples, leading to steep gradients and potential oscillation of the generated data around the real data.

Towards addressing the threat of model oscillation, tuning α_D below 1 slightly increases (decreases) the gradient scalar $C^{\mathrm{NS}}_{\alpha_D,\alpha_G}$ received by the generated samples close to (far from) the real modes, as shown in Figure 5(b). Intuitively, this strategy incentivizes the generator to preserve the proximity of its generated samples to the real samples, even if there exist outlier samples with steep gradients. As a result, the generated samples are robust to outliers and therefore more likely to converge to the real modes. In Figure 6(b), we observe that the non-saturating generator loss with $\alpha_D < 1$ possesses a meaningful gradient in the dense p_r region, while significantly reducing the steep gradient in the dense p_{G_θ} region. We also find that tuning α_G above 1 enhances the same optimization strategy – which is consistent with analysis on the insensitivity of α-loss to outliers (Sypherd et al. 2019) – but the convergence of $C^{\mathrm{NS}}_{\alpha_D,\alpha_G}$ to zero when $P(Y = 1|X) \to 0$ can be problematic since the near-zero gradients may immobilize data far from the real distribution.

Chapter 4

EXPERIMENTAL RESULTS

In this section, we present empirical evidence for the effectiveness of (α_D, α_G)-GAN compared to both the vanilla GAN (i.e., the $(1,1)$-GAN) and the state-of-the-art Least Squares GAN (LSGAN) (Mao et al. 2017). We conduct evaluations [2] on three datasets: (i) a synthetic dataset generated by a two-dimensional, ring-shaped Gaussian mixture distribution (2D-ring) (Srivastava et al. 2017), (ii) the Celeb-A image dataset with dimensions of 64×64 (Liu et al. 2015), and (iii) the LSUN Classroom image dataset with dimensions of 112×112 (Yu et al. 2015). For each dataset and type of GAN, we report multiple metrics that capture the stability of GAN training across numerous random seeds. This analysis serves to highlight the potential of tuning (α_D, α_G) to achieve stable and robust solutions for image generation.

4.1 2D Gaussian Mixture Ring

The 2D-ring is an oft-used synthetic dataset for evaluating GANs. We draw samples from a mixture of 8 equal-prior Gaussian distributions, indexed $i \in \{1, 2, \ldots, 8\}$ with a mean of $(\cos(2\pi i/8),\ \sin(2\pi i/8))$ and variance 10^{-4}. We generate 50,000 training and 25,000 testing samples; additionally, we generate the same number of 2D latent Gaussian noise vectors. Both the discriminator and generator networks have 4 fully-connected layers with 200 and 400 units, respectively. We train for 400 epochs with a batch size of 128, and optimize with Adam (Kingma and Ba 2014) and a learning

[2] Our repository can be found at https://github.com/SankarLab/AlphaGan-Papers-Results

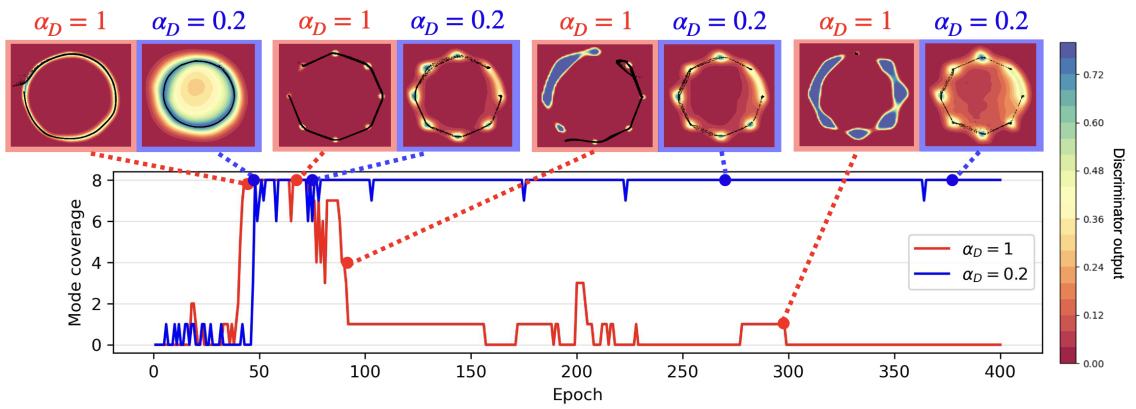
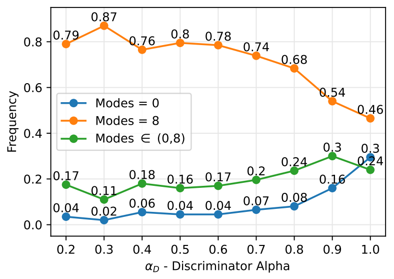


Figure 7. Plot of mode coverage over epochs for saturating $(\alpha_D, 1)$-GAN with $\alpha_D \in \{0.2, 1\}$. Placed above this plot are 2D visuals of the generated samples (in black) at different epochs, and the discriminator outputs are illustrated as heat maps.

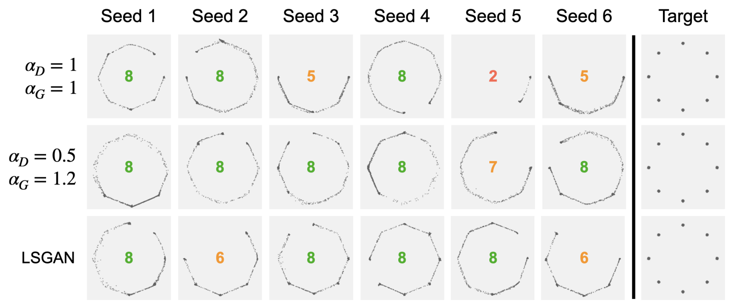
rate of 10^{-4} for both models. We consider three types of GANs that differ in their objectives: **(i)** saturating (α_D, α_G)-GAN in (3.4); **(ii)** non-saturating (α_D, α_G)-GAN in (3.4a), (3.6); **(iii)** LSGAN with the 0-1 binary coding scheme (see Appendix B.1).

For every setting listed above, we train our models on the 2D-ring dataset for 200 random state seeds, where each seed contains different weight initializations for the discriminator and generator. Ideally, a stable method will reflect similar performance across randomized initializations and also over training epochs; thus, we explore how GAN training performance for each setting varies across seeds and epochs. Our primary performance metric is *mode coverage*, defined as the number of Gaussians (0-8) that contain a generated sample within 3 standard deviations of its mean. A score of 8 conveys successful training, while a score of 0 conveys a significant GAN failure; on the other hand, a score in between 0 and 8 may be indicative of common GAN issues, such as mode collapse or failure to converge.

For the saturating setting, the improvement in stability of the $(0.2, 1)$-GAN relative to the vanilla GAN is illustrated in Figure 7. Specifically, we observe that



(a)



(b)

Figure 8. (a) Plot of success and failure rates over 200 seeds for the saturating $(\alpha_D, 1)$-GAN with $\alpha_D \in [0.2, 1]$ trained on the 2D-ring dataset. (b) Generated 2D-ring samples from the vanilla GAN, $(0.5, 1.2)$-GAN, and LSGAN over 6 seeds.

both GANs successfully capture the ring-like structure, but the vanilla GAN fails to maintain the ring over time. This can be attributed to the overconfident nature of the $\alpha_D = 1$ discriminator, which in turn subjects the generator to exploding and vanishing gradients. On the other hand, the less confident predictions of the $(0.2, 1)$-GAN create a smooth landscape for the generated output to descend towards the real data.

In general, Figure 8(a) shows that the vanilla GAN completely fails to recover the true distribution 30% of the time, while succeeding only 46% of the time. Conversely, the $(\alpha_D, 1)$-GAN with $\alpha_D < 1$ learns a more stable generator due to a less confident discriminator; for example, the $(0.3, 1)$-GAN success and failure rates improve to 87% and 2%, respectively. Lastly, for the non-saturating setting in Figure 8(b), we find that tuning $\alpha_D < 1$ and $\alpha_G > 1$ consistently yields more stable outcomes than do vanilla GANs and LSGANs. Mode coverage rates over 200 seeds for saturating (Tables 2 and 3) and non-saturating (Table 4) are in Appendix B.

4.2 Celeb-A & LSUN Classroom

The Celeb-A dataset (Liu et al. 2015) is a widely recognized large-scale collection of over 200,000 celebrity headshots, encompassing images with diverse aspect ratios, camera angles, backgrounds, lighting conditions, and other variations. Similarly, the LSUN Classroom dataset (Yu et al. 2015) is a subset of the comprehensive Large-scale Scene Understanding (LSUN) dataset; it contains over 150,000 classroom images captured under diverse conditions and with varying aspect ratios. To ensure consistent input for the discriminator, we follow the standard practice of resizing the images to 64×64 for Celeb-A and 112×112 for LSUN Classroom. For both experiments, we randomly select 80% of the images for training and leave the remaining 20% for validation (evaluation of goodness metrics). Lastly, for each dataset, we generate a similar 80%-20% training-validation split of 100-dimensional latent Gaussian noise vectors, for a total matching the size of the true data.

For training, we employ the DCGAN architecture (Radford, Metz, and Chintala

GAN	Celeb-A								LSUN Classroom				
	Learning rate ($\times 10^{-4}$)												
(α_d, α_g)	1	2	5	6	7	8	9	10	1	2	3	4	5
$(1, 1)$	**100**	93	83	60	59	39	54	55	92	36	13	13	12
$(0.9, 1)$	**100**	95	78	72	81	67	74	47	76	53	22	17	22
$(0.8, 1)$	98	**98**	**89**	82	81	72	68	76	89	61	36	28	29
$(0.7, 1)$	**100**	91	**89**	**92**	**86**	**81**	68	**80**	90	80	78	67	55
$(0.6, 1)$	98	93	88	77	85	76	**77**	69	**96**	**90**	**85**	**78**	**66**
$(0.5, 1)$	93	76	14	8	5	4	2	0	94	77	78	37	26

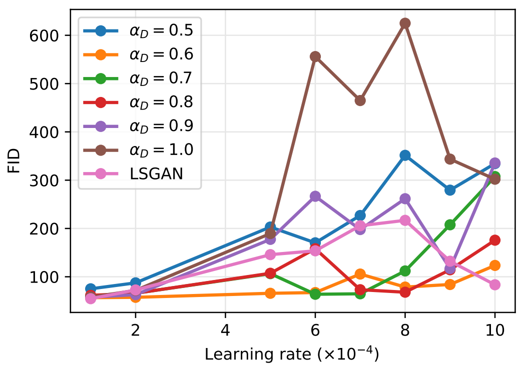
Table 1. Percentage out of 50 seeds of FID scores below 80 (Celeb-A) or 800 (LSUN Classroom) for each combination of (α_D, α_G)-GAN and learning rate, trained for 100 epochs. Best results for each dataset and learning rate are emboldened.

2015) that leverages deep convolutional neural networks (CNNs) for both the discriminator and generator. In Appendix B, detailed descriptions of the discriminator and generator architectures can be found in Table 5 for Celeb-A and Table 6 for LSUN Classroom. Following previous works, our focus is solely on the non-saturating setting, utilizing appropriate objective functions for vanilla GAN, (α_D, α_G)-GAN, and LSGAN. We consider a variety of learning rates, ranging from 10^{-4} to 10^{-3}, for Adam optimization. Additionally, we evaluate our models every 10 epochs up to a total of 100 epochs; in doing so, we report the Fréchet Inception Distance (FID), an unsupervised similarity metric between the real and generated feature distributions extracted by InceptionNet-V3 (Heusel et al. 2017). For both datasets, we train each combination of objective function, number of epochs, and learning rate for 50 seeds.

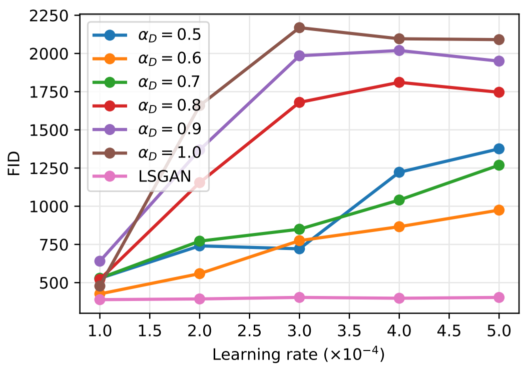
In the following subsections, we empirically demonstrate the dependence of inception distance on learning rate and number of epochs for the vanilla GAN, (α_D, α_G)-GAN, and LSGAN. Achieving robustness to hyperparameter initialization is especially desirable in the unsupervised GAN setting as the choices that facilitate steady model convergence are not easily determined without prior mode knowledge.

4.2.1 Celeb-A Results

In Figure 9(a), we examine the relationship between learning rate and FID for each GAN trained for 100 epochs on the Celeb-A dataset. When using learning rates of 1×10^{-4} and 2×10^{-4}, all GANs consistently perform well. For instance, Table 1 demonstrates that GANs with a learning rate of 1×10^{-4} achieve an FID score below 80 at least 93% of the time. However, when the learning rate increases to 5×10^{-4}, the vanilla $(1, 1)$-GAN begins to exhibit instability across the 50 seeds. This is evident



(a)



(b)

Figure 9. (a) Plot of Celeb-A FID scores averaged over 50 seeds for 8 different learning rates and 7 different GANs trained for 100 epochs. (b) Plot of LSUN Classroom FID scores averaged over 50 seeds for 5 different learning rates and 7 different GANs trained for 100 epochs.

in Table 1, where the $(1, 1)$-GAN achieves an FID score below 80 only 82.6% of the time. As the learning rate surpasses 5×10^{-4}, the performance of the vanilla GAN becomes even more erratic, underscoring the importance of GANs being robust to the choice of learning rate.

Interestingly, we observe that tuning α_D below 1 contributes to stabilizing the FID scores over the 50 seeds while maintaining relatively low scores on average. For instance, in Figure 9, we can see that the $(0.6, 1)$-GAN consistently achieves low FIDs across all tested learning rates. Moreover, Table 1 emphasizes the stability of the $(0.7, 1)$-GAN, as it achieves an FID score below 80 at least 80% of the time for 7 of the learning rates. Comparatively, Figure 9(a) demonstrates that the GANs with $\alpha_D < 1$ perform on par with, if not better than, the state-of-the-art LSGAN. Additionally, in Figure 11(a), we compare the learning rate sensitivities of the vanilla $(1, 1)$-GAN, $(0.6, 1)$-GAN, and LSGAN by plotting their FIDs across 10 steps of 100 epochs for two similar learning rates: 1×10^{-4} and 2×10^{-4}. We discover that the vanilla $(1, 1)$-GAN performs significantly worse for the higher learning rate and deteriorates over time

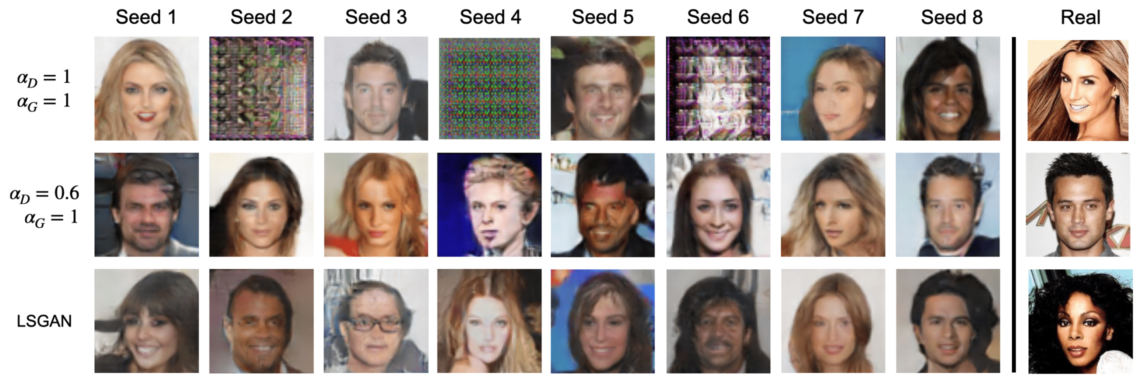


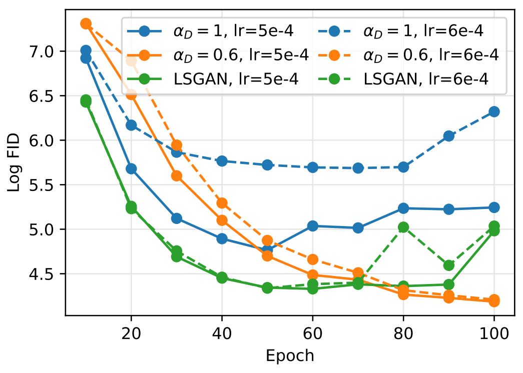
Figure 10. Generated Celeb-A faces from the same three GANs over 8 seeds when trained for 100 epochs with a learning rate of 5×10^{-4}.

for both learning rates. Conversely, both the $(0.6, 1)$-GAN and LSGAN consistently exhibit favorable FID performance for both learning rates. However, the $(0.6, 1)$-GAN converges to a low FID, while the FID of the LSGAN slightly increases as training approaches 100 epochs.

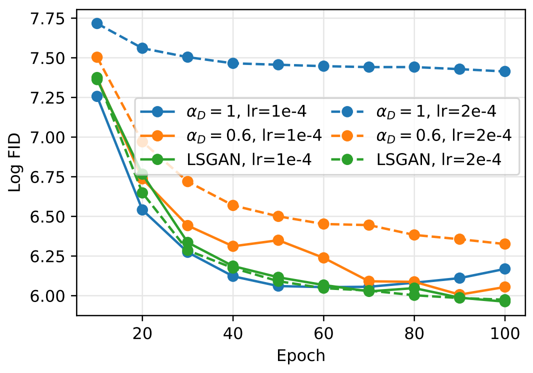
Lastly, Figure 10 displays a grid of generated Celeb-A faces, randomly sampled over 8 seeds for three GANs trained for 100 epochs with a learning rate of 5×10^{-4}. Here, we observe that the faces generated by the $(0.6, 1)$-GAN and LSGAN exhibit a comparable level of quality to the rightmost column images, which are randomly sampled from the real Celeb-A dataset. On the other hand, the vanilla $(1, 1)$-GAN shows clear signs of performance instability, as some seeds yield high-quality images while others do not.

4.2.2 LSUN Classroom Results

Figure 9(a) illustrates the relationship between learning rate and FID for GANs trained on the LSUN dataset for 100 epochs. First, we observe that when all GANs



(a)



(b)

Figure 11. (a) Log-scale plot of Celeb-A FID scores over training epochs in steps of 10 up to 100 total, for three noteworthy GANs– $(1, 1)$-GAN (vanilla), $(0.6, 1)$-GAN, and LSGAN– and for two similar learning rates– 5×10^{-4} and 6×10^{-4}. (b) Log-scale plot of LSUN Classroom FID scores over training epochs in steps of 10 up to 100 total, for three noteworthy GANs– $(1, 1)$-GAN (vanilla), $(0.6, 1)$-GAN, and LSGAN– and for two similar learning rates– 1×10^{-4} and 2×10^{-4}.

train with a learning rate of 1×10^{-4}, they consistently deliver satisfactory performance. For instance, as shown in Table 1, GANs with a learning rate of 1×10^{-4} achieve an FID score below 800 in at least 76% of the cases. However, raising the learning rate to 2×10^{-4} leads to instability in the vanilla $(1, 1)$-GAN across the 50 seeds; this trend is reflected in Table 1, where the $(1, 1)$-GAN achieves an FID score below 800 only 36.2% of the time. As the learning rate exceeds 2×10^{-4}, the performance stability of the vanilla GAN diminishes even further, plummeting to as low as 12.2% when training with a learning rate of 5×10^{-4}.

However, we once again observe that reducing α_D below 1 contributes to stabilizing the FID across the 50 seeds when trained with slightly higher learning rates. In Figure 9(b), we see that as α_D is tuned down to 0.6, the mean FIDs consistently decrease across all tested learning rates. These lower FIDs can be attributed to the increased stability of the network, as indicated in Table 1. For example, the $(0.6, 1)$-GAN achieves an FID score below 800 at least two-thirds of the time when trained with the

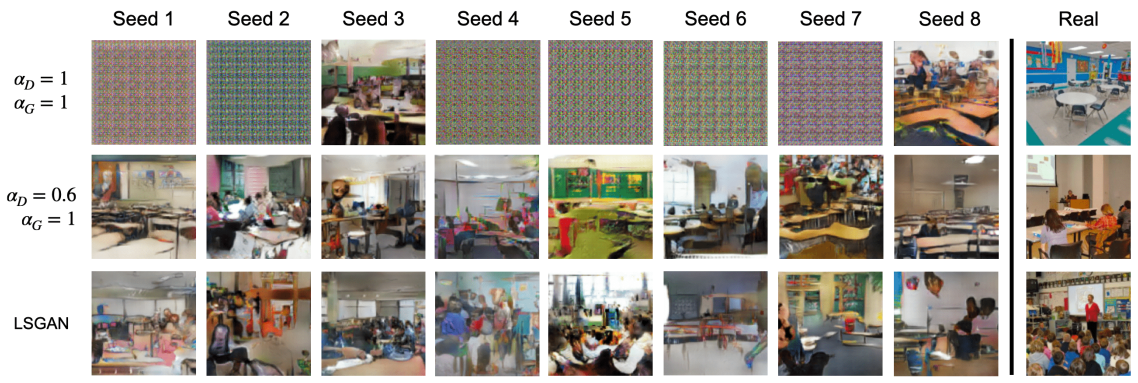


Figure 12. Generated LSUN Classroom images from the same three GANs over 8 seeds when trained for 100 epochs with a learning rate of 2×10^{-4}.

highest learning rate. Despite the gains in GAN stability achieved by tuning down α_D, Figure 9 still demonstrates a noticeable disparity between the best (α_D, α_G)-GAN and the state-of-the-art LSGAN. This suggests that there is still room for improvement in generating high-dimensional images with (α_D, α_G)-GANs.

Furthermore, Figure 11(b) illustrates the average FID throughout the training process for three GANs: $(1, 1)$-GAN, $(0.6, 1)$-GAN, and LSGAN, using two different learning rates. These findings validate that the vanilla $(1, 1)$-GAN performs well when trained with the lower learning rate, but struggles significantly with the higher learning rate. In contrast, the $(0.6, 1)$-GAN shows less sensitivity to the choice of learning rate, while the LSGAN achieves nearly identical scores for both learning rates. To showcase the image quality generated by each GAN at epoch 100 with the higher learning rate, refer to Figure 12. It is evident from the figure that the vanilla $(1, 1)$-GAN frequently fails during training, whereas the $(0.6, 1)$-GAN and LSGAN produce images that closely resemble the real distribution.

Chapter 5

CONCLUSION

Overall, this thesis has explored the challenges faced by the vanilla GAN, and introduced the (α_D, α_G)-GAN architecture as a solution to improve their training stability and quality of generated samples. The objectives of vanilla GANs, while intuitive, often lead to issues such as exploding and vanishing gradients, mode collapse, and model oscillation. Alternative GAN architectures have been proposed but have shown limited success in overcoming these challenges.

The (α_D, α_G)-GAN formulation addresses these limitations by allowing the tuning of parameters α_D and α_G to achieve a more stable training process. The specific objectives for the generator and discriminator in (α_D, α_G)-GAN have been derived and analyzed through gradient analysis. Experimental results conducted on various datasets – including 2D Gaussian Mixture Ring, Celeb-A, and LSUN Classroom – have demonstrated the effectiveness and stability of (α_D, α_G)-GAN compared to the vanilla GAN and LSGAN. Tuning α_D below 1 has shown robustness to hyperparameter choices and achieved competitive performance compared to LSGAN.

By providing both theoretical insights and empirical evidence, this research contributes to the advancement of GANs and offers valuable insights for generating more realistic synthetic data. The (α_D, α_G)-GAN architecture presents a promising approach to address the failures of vanilla GANs and improve the training stability and quality of generated samples. Future research can further explore and refine the (α_D, α_G)-GAN architecture, potentially leading to even more significant advancements in the field of generative models.

REFERENCES

Arjovsky, Martin, and Léon Bottou. 2017. “Towards principled methods for training generative adversarial networks.” *arXiv preprint arXiv:1701.04862.*

Bhatia, Himesh, William Paul, Fady Alajaji, Bahman Gharesifard, and Philippe Burlina. 2021. “Least kth-Order and Rényi Generative Adversarial Networks.” *Neural Computation* 33 (9): 2473–2510.

Donahue, Chris, Julian J. McAuley, and Miller S. Puckette. 2018. “Synthesizing Audio with Generative Adversarial Networks.” *CoRR* abs/1802.04208. arXiv: 1802.04208. http://arxiv.org/abs/1802.04208.

Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. 2015. “A Neural Algorithm of Artistic Style.” *CoRR* abs/1508.06576. arXiv: 1508.06576. http://arxiv.org/abs/1508.06576.

Goodfellow, Ian J., Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. “Generative Adversarial Nets.” In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2,* 2672–2680.

Heusel, Martin, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Günter Klambauer, and Sepp Hochreiter. 2017. “GANs Trained by a Two Time-Scale Update Rule Converge to a Nash Equilibrium.” *arXiv preprint arXiv:1706.08500.*

Jiang, Yifan, Han Chen, Murray Loew, and Hanseok Ko. 2020. “COVID-19 CT image synthesis with a conditional generative adversarial network.” *IEEE Journal of Biomedical and Health Informatics* 25 (2): 441–452.

Kingma, Diederik P, and Jimmy Ba. 2014. “Adam: A method for stochastic optimization.” *arXiv preprint arXiv:1412.6980.*

Kurri, Gowtham R., Tyler Sypherd, and Lalitha Sankar. 2021. “Realizing GANs via a Tunable Loss Function.” In *IEEE Information Theory Workshop (ITW),* 1–6.

Kurri, Gowtham R., Monica Welfert, Tyler Sypherd, and Lalitha Sankar. 2022. “α-GAN: Convergence and Estimation Guarantees.” In *IEEE International Symposium on Information Theory (ISIT),* 276–281.

Liese, F., and I. Vajda. 2006. “On Divergences and Informations in Statistics and Information Theory.” *IEEE Transactions on Information Theory* 52 (10): 4394–4412.

Liu, Ziwei, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. "Deep Learning Face Attributes in the Wild." In *Proceedings of International Conference on Computer Vision (ICCV).* December.

Mao, Xudong, Qing Li, Haoran Xie, Raymond Y.K. Lau, Zhen Wang, and Stephen Paul Smolley. 2017. "Least Squares Generative Adversarial Networks." In *Proceedings of the IEEE International Conference on Computer Vision (ICCV).*

Nowozin, Sebastian, Botond Cseke, and Ryota Tomioka. 2016. "f-GAN: Training Generative Neural Samplers Using Variational Divergence Minimization." In *Proceedings of the 30th International Conference on Neural Information Processing Systems,* 271–279.

Österreicher, Ferdinand. 1996. "On a class of perimeter-type distances of probability distributions." *Kybernetika* 32 (4): 389–393.

Radford, Alec, Luke Metz, and Soumith Chintala. 2015. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks." *arXiv preprint arXiv:1511.06434.*

Srivastava, Akash, Lazar Valkov, Chris Russell, Michael U. Gutmann, and Charles Sutton. 2017. "VEEGAN: Reducing Mode Collapse in GANs using Implicit Variational Learning." In *Advances in Neural Information Processing Systems,* vol. 30.

Sypherd, Tyler, Mario Diaz, John Kevin Cava, Gautam Dasarathy, Peter Kairouz, and Lalitha Sankar. 2022. "A Tunable Loss Function for Robust Classification: Calibration, Landscape, and Generalization." *IEEE Transactions on Information Theory* 68 (9): 6021–6051.

Sypherd, Tyler, Mario Diaz, Lalitha Sankar, and Peter Kairouz. 2019. "A tunable loss function for binary classification." In *IEEE International Symposium on Information Theory,* 2479–2483.

Wiatrak, Maciej, Stefano V Albrecht, and Andrew Nystrom. 2019. "Stabilizing Generative Adversarial Networks: A Survey." *arXiv preprint arXiv:1910.00927.*

Yu, Fisher, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. 2015. "LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop." *CoRR* abs/1506.03365. arXiv: 1506.03365. http://arxiv.org/abs/1506.03365.

APPENDIX A

THEOREM PROOFS

A.1 Proof of Theorem 3.2.1

Proof. We first begin with the optimal discriminator equation in (3.7):

$$D_{\omega^*}(x) = \frac{p_r(x)^{\alpha_D}}{p_r(x)^{\alpha_D} + p_{G_\theta}(x)^{\alpha_D}}. \tag{A.1}$$

Then we do some regrouping

$$= \frac{\frac{p_r(x)^{\alpha_D}}{\left(p_r(x)+p_{G_\theta}(x)\right)^{\alpha_D}}}{\frac{p_r(x)^{\alpha_D}}{\left(p_r(x)+p_{G_\theta}(x)\right)^{\alpha_D}} + \frac{p_{G_\theta}(x)^{\alpha_D}}{\left(p_r(x)+p_{G_\theta}(x)\right)^{\alpha_D}}} \tag{A.2}$$

$$= \frac{\left(\frac{p_r(x)^{\alpha_D}}{p_r(x)+p_{G_\theta}(x)}\right)^{\alpha_D}}{\left(\frac{p_r(x)^{\alpha_D}}{p_r(x)+p_{G_\theta}(x)}\right)^{\alpha_D} + \left(\frac{p_{G_\theta}(x)^{\alpha_D}}{p_r(x)+p_{G_\theta}(x)}\right)^{\alpha_D}} \tag{A.3}$$

and substitute via equation (3.9)

$$= \frac{P(Y=1|X=x)^{\alpha_D}}{P(Y=1|X=x)^{\alpha_D} + P(Y=0|X=x)^{\alpha_D}} \tag{A.4}$$

$$= P_{\alpha_D}(Y=1|X=x). \tag{A.5}$$

□

A.2 Proof of Theorem 3.2.2

Proof. Suppose the discriminator D_{ω^*} is fixed and optimizes the sup of $V_{\alpha_D}(\theta, \omega)$. Then we know from (3.7) that

$$D_{\omega^*}(x) = \frac{p_r(x)^{\alpha_D}}{p_r(x)^{\alpha_D} + p_{G_\theta}(x)^{\alpha_D}}. \tag{A.6}$$

First we derive $\partial D_{\omega^*}/\partial x$ using the quotient rule, which will be helpful later on.

$$\begin{aligned}
\frac{\partial D_{\omega^*}}{\partial x} &= (p_r(x)^{\alpha_D} + p_{G_\theta}(x)^{\alpha_D})^{-2}\,[(p_r(x)^{\alpha_D} + p_{G_\theta}(x)^{\alpha_D})(\alpha_D p_r(x)^{\alpha_D - 1}\frac{\partial p_r}{\partial x}) \\
&\quad - (p_r(x)^{\alpha_D})(\alpha_D p_r(x)^{\alpha_D-1}\frac{\partial p_r}{\partial x} + \alpha_D p_{G_\theta}(x)^{\alpha_D-1}\frac{\partial p_{G_\theta}}{\partial x})] \qquad \text{(A.7)} \\
&= C_1[\alpha_D p_r(x)^{2\alpha_D-1}\frac{\partial p_r}{\partial x} + \alpha_D p_r(x)^{\alpha_D-1} p_{G_\theta}(x)^{\alpha_D}\frac{\partial p_r}{\partial x} \\
&\quad - \alpha_D p_r(x)^{2\alpha_D-1}\frac{\partial p_r}{\partial x} - \alpha_D p_r(x)^{\alpha_D} p_{G_\theta}(x)^{\alpha_D-1}\frac{\partial p_{G_\theta}}{\partial x}] \qquad \text{(A.8)} \\
&\quad \text{where } C_1 = (p_r(x)^{\alpha_D} + p_{G_\theta}(x)^{\alpha_D})^{-2} \\
&= C_1\left(\alpha_D p_r(x)^{\alpha_D-1} p_{G_\theta}(x)^{\alpha_D}\frac{\partial p_r}{\partial x} - \alpha_D p_r(x)^{\alpha_D} p_{G_\theta}(x)^{\alpha_D-1}\frac{\partial p_{G_\theta}}{\partial x}\right) \qquad \text{(A.9)} \\
&= C_2\left(\frac{1}{p_r(x)}\frac{\partial p_r}{\partial x} - \frac{1}{p_{G_\theta}(x)}\frac{\partial p_{G_\theta}}{\partial x}\right) \qquad \text{(A.10)} \\
&\quad \text{where } C_2 = \alpha_D p_r(x)^{\alpha_D} p_{G_\theta}(x)^{\alpha_D}\,(p_r(x)^{\alpha_D} + p_{G_\theta}(x)^{\alpha_D})^{-2} \\
&\quad \text{which simplifies to } C_2 = \alpha_D D_{\omega^*}(x)\,(1 - D_{\omega^*}(x)) \\
&= \alpha_D D_{\omega^*}(x)\,(1 - D_{\omega^*}(x))\left(\frac{1}{p_r(x)}\frac{\partial p_r}{\partial x} - \frac{1}{p_{G_\theta}(x)}\frac{\partial p_{G_\theta}}{\partial x}\right) \qquad \text{(A.11)}
\end{aligned}$$

Next, we set $\mu = D_{\omega^*}(x)$ and derive $-\partial \ell_{\alpha_G}(0,\mu)/\partial\mu$ as follows

$$\begin{aligned}
-\frac{\partial \ell_{\alpha_G}(0,\mu)}{\partial \mu} &= \frac{\partial}{\partial \mu}\left[-\frac{\alpha_G}{\alpha_G - 1}\left(1 - (1-\mu)^{1-1/\alpha_G}\right)\right] \qquad \text{(A.12)} \\
&= -(1-\mu)^{-1/\alpha_G}. \qquad \text{(A.13)}
\end{aligned}$$

Lastly, to find the gradient $-\partial \ell_{\alpha_G}(0, D_{\omega^*}(x))/\partial x$, we apply the chain rule and substitute via equations (A.11), (A.13):

$$\begin{aligned}
-\frac{\partial \ell_{\alpha_G}(0, D_{\omega^*}(x))}{\partial x} &= -\frac{\partial \ell_{\alpha_G}(0, D_{\omega^*}(x))}{\partial D_{\omega^*}} \times \frac{\partial D_{\omega^*}}{\partial x} \qquad \text{(A.14)} \\
&= C_{\alpha_D,\alpha_G}\left(\frac{1}{p_{G_\theta}(x)}\frac{\partial p_{G_\theta}}{\partial x} - \frac{1}{p_r(x)}\frac{\partial p_r}{\partial x}\right) \qquad \text{(A.15)} \\
&\quad \text{where } C_{\alpha_D,\alpha_G} = \alpha_D D_{\omega^*}(x)\,(1 - D_{\omega^*}(x))^{1-1/\alpha_G} \\
&\quad \text{or equivalently,} \\
&\quad C_{\alpha_D,\alpha_G} = \alpha_D P_{\alpha_D}(Y=1|X=x)\,(1 - P_{\alpha_D}(Y=1|X=x))^{1-1/\alpha_G}.
\end{aligned}$$

Since the scalar C_{α_D,α_G} is positive and the only term reliant on α_D and α_G, we conclude that the direction of $-\partial \ell_{\alpha_G}(0, D_{\omega^*}(x))/\partial x$ is independent of these parameters. □

A.3 Proof of Theorem 3.2.3

Proof. This result will follow similarly to Theorem 3.2.2. First, we set $\mu = D_{\omega^*}(x)$ and derive $\partial \ell_{\alpha_G}(1,\mu)/\partial \mu$ as follows:

$$\begin{aligned}\frac{\partial \ell_{\alpha_G}(1,\mu)}{\partial \mu} &= \frac{\partial}{\partial \mu}\left[\frac{\alpha_G}{\alpha_G - 1}\left(1 - \mu^{1-1/\alpha_G}\right)\right] && \text{(A.16)}\\ &= -\mu^{-1/\alpha_G} && \text{(A.17)}\end{aligned}$$

Then we derive the gradient $\partial \ell_{\alpha_G}(1, D_{\omega^*}(x))/\partial x$ with the chain rule by substituting via equations (A.13), (A.17):

$$\begin{aligned}\frac{\partial \ell_{\alpha_G}(1, D_{\omega^*}(x))}{\partial x} &= \frac{\partial \ell_{\alpha_G}(1, D_{\omega^*}(x))}{\partial D_{\omega^*}} \times \frac{\partial D_{\omega^*}}{\partial x} && \text{(A.18)}\\ &= C^{\text{NS}}_{\alpha_D,\alpha_G}\left(\frac{1}{p_{G_\theta}(x)}\frac{\partial p_{G_\theta}}{\partial x} - \frac{1}{p_r(x)}\frac{\partial p_r}{\partial x}\right) && \text{(A.19)}\\ &\text{where } C^{\text{NS}}_{\alpha_D,\alpha_G} = \alpha_D\left(1 - D_{\omega^*}(x)\right) D_{\omega^*}(x)^{1-1/\alpha_G}\\ &\text{or equivalently,}\\ &C_{\alpha_D,\alpha_G} = \alpha_D\left(1 - P_{\alpha_D}(Y=1|X=x)\right) P_{\alpha_D}(Y=1|X=x)^{1-1/\alpha_G}.\end{aligned}$$

Since the scalar $C^{\text{NS}}_{\alpha_D,\alpha_G}$ is positive and the only term reliant on α_D and α_G, we conclude that the direction of $\partial \ell_{\alpha_G}(1, D_{\omega^*}(x))/\partial x$ is independent of these parameters.

□

APPENDIX B

EXPERIMENT DETAILS

B.1 Brief Overview of LSGAN

The Least Squares GAN (LSGAN) is a dual-objective, min-max game introduced in (Mao et al. 2017). The LSGAN objective functions, as the name suggests, involve squared loss functions for the generator and discriminator which are written as

$$\begin{aligned} &\inf_{\omega\in\Omega} \frac{1}{2}\Big(\mathbb{E}_{X\sim P_r}[(D_\omega(X)-b)^2]+\mathbb{E}_{X\sim P_{G_\theta}}[(D_\omega(X)-a)^2]\Big) \\ &\inf_{\theta\in\Theta} \frac{1}{2}\Big(\mathbb{E}_{X\sim P_r}[(D_\omega(X)-c)^2]+\mathbb{E}_{X\sim P_{G_\theta}}[(D_\omega(X)-c)^2]\Big). \end{aligned} \tag{B.1}$$

For appropriately chosen values of the parameters a, b, and c, (B.1) reduces to minimizing the Pearson χ^2-divergence between $P_r + P_{G_\theta}$ and $2P_{G_\theta}$. As done in the original paper (Mao et al. 2017), we use $a = 0$, $b = 1$ and $c = 1$ for our experiments to make fair comparisons. The authors refer to this choice of parameters as the 0-1 binary coding scheme.

Table 2. Success rates for 2D-ring with the saturating (α_D, α_G)-GAN over 200 seeds, with top 4 combinations emboldened.

% of success		α_D					
(8/8 modes)		0.5	0.6	0.7	0.8	0.9	1.0
α_G	0.9	73	**79**	69	60	46	34
	1.0	**80**	**79**	74	68	54	47
	1.1	**79**	77	68	70	59	47
	1.2	75	74	71	65	57	46

Table 3. Failure rates for 2D-ring with the saturating (α_D, α_G)-GAN over 200 seeds, with top 3 combinations emboldened.

% of failure		α_D					
(0/8 modes)		0.5	0.6	0.7	0.8	0.9	1.0
α_G	0.9	11	10	12	13	29	49
	1.0	**5**	**5**	7	8	16	30
	1.1	7	9	13	12	13	26
	1.2	9	**5**	9	12	17	31

B.2 2D Gaussian Mixture Ring

In Tables 2 and 3, we report the success (8/8 mode coverage) and failure (0/8 mode coverage) rates over 200 seeds for a grid of (α_D, α_G) combinations for the *saturating* setting. Compared to the vanilla GAN performance, we find that tuning α_D below 1 leads to a greater success rate and lower failure rate. However, in this setting, we find that tuning α_G away from 1 has no significant impact on GAN performance.

In Table 4, we detail the success rates for the *non-saturating* setting. We note that for this dataset, no failures – and therefore, no vanishing/exploding gradients – occurred in this setting. In particular, we find that the $(0.5, 1.2)$-GAN doubles the success rate of the vanilla $(1, 1)$-GAN, which is more susceptible to mode collapse as illustrated in Figure 8(b). We also find that LSGAN achieves a success rate of 32.5%, which is greater than vanilla GAN but less than the best-performing (α_D, α_G)-GAN.

Table 4. Success rates for 2D-ring with the non-saturating (α_D, α_G)-GAN over 200 seeds, with top 5 combinations emboldened.

% of success		α_D							
(8/8 modes)		0.5	0.6	0.7	0.8	0.9	1.0	1.1	1.2
α_G	0.8	35	24	19	19	14	16	18	10
	0.9	**39**	37	19	22	16	20	19	21
	1.0	34	35	29	28	26	22	20	32
	1.1	**40**	36	31	22	24	15	23	25
	1.2	**45**	38	34	25	26	28	20	22
	1.3	**44**	**39**	26	28	28	25	31	29

B.3 Celeb-A & LSUN Classroom

The discriminator and generator architectures used for the Celeb-A and LSUN Classroom datasets are described in Tables 5 and 6 respectively. Each architecture consists of four convolution layers, with parameters such as kernel size (a.k.a., filter size), stride (the amount by which the filter moves), and the activation functions applied to the layer outputs. Zero padding is also assumed. In both tables, "BN" represents batch normalization, a technique that normalizes the inputs to each layer using a batch of samples during model training. Batch normalization is commonly employed in deep learning to prevent cumulative floating point errors and overflows, and to ensure that all features remain within a similar range. This technique serves as a computational tool to address vanishing and/or exploding gradients.

Table 5. Discriminator and generator architectures for Celeb-A dataset.

Discriminator					
Layer	Output size	Kernel	Stride	BN	Activation
Input	$3 \times 64 \times 64$				Leaky ReLU
Convolution	$64 \times 32 \times 32$	4×4	2	Yes	Leaky ReLU
Convolution	$128 \times 16 \times 16$	4×4	2	Yes	Leaky ReLU
Convolution	$256 \times 8 \times 8$	4×4	2	Yes	Leaky ReLU
Convolution	$512 \times 4 \times 4$	4×4	2	Yes	Leaky ReLU
Convolution	$1 \times 1 \times 1$	4×4	2		Sigmoid
Generator					
Layer	Output size	Kernel	Stride	BN	Activation
Input	$100 \times 1 \times 1$				ReLU
ConvTranspose	$512 \times 4 \times 4$	4×4	2	Yes	ReLU
ConvTranspose	$256 \times 8 \times 8$	4×4	2	Yes	ReLU
ConvTranspose	$128 \times 16 \times 16$	4×4	2	Yes	ReLU
ConvTranspose	$64 \times 32 \times 32$	4×4	2	Yes	ReLU
ConvTranspose	$3 \times 64 \times 64$	4×4	2		Tanh

Table 6. Discriminator and generator architectures for LSUN Classroom dataset.

Discriminator					
Layer	Output size	Kernel	Stride	BN	Activation
Input	$3 \times 112 \times 112$				Leaky ReLU
Convolution	$64 \times 56 \times 56$	4×4	2	Yes	Leaky ReLU
Convolution	$128 \times 28 \times 28$	4×4	2	Yes	Leaky ReLU
Convolution	$256 \times 14 \times 14$	4×4	2	Yes	Leaky ReLU
Convolution	$512 \times 7 \times 7$	4×4	2	Yes	Leaky ReLU
Convolution	$1 \times 1 \times 1$	7×7	2		Sigmoid
Generator					
Layer	Output size	Kernel	Stride	BN	Activation
Input	$100 \times 1 \times 1$				ReLU
ConvTranspose	$512 \times 7 \times 7$	7×7	2	Yes	ReLU
ConvTranspose	$256 \times 14 \times 14$	4×4	2	Yes	ReLU
ConvTranspose	$128 \times 28 \times 28$	4×4	2	Yes	ReLU
ConvTranspose	$64 \times 56 \times 56$	4×4	2	Yes	ReLU
ConvTranspose	$3 \times 112 \times 112$	4×4	2		Tanh