FAKE CAN BE REAL IN GANS

Anonymous authors
Paper under double-blind review

ABSTRACT

In order to alleviate the notorious mode collapse phenomenon in generative adversarial networks (GANs), we propose a novel training method of GANs in which certain fake samples can be reconsidered as real ones during the training process. This strategy can reduce the gradient value that generator receives in the region where gradient exploding happens. We show that the theoretical equilibrium between the generators and discriminations actually can be seldom realized in practice. And this results in an unbalanced generated distribution that deviates from the target one, when fake datapoints overfit to real ones, which explains the non-stability of GANs. We also prove that, by penalizing the difference between discriminator outputs and considering certain fake datapoints as real for adjacent real and fake sample pairs, gradient exploding can be alleviated. Accordingly, a modified GAN training method is proposed with a more stable training process and a better generalization. Experiments on different datasets verify our theoretical analysis.

1 INTRODUCTION

In the past few years, Generative Adversarial Networks (GANs) have been one of the most popular topics in generative models and achieved great success in generating diverse and high-quality images recently (Brock et al. (2019); Karras et al. (2019); Donahue & Simonyan (2019)). GANs are powerful tools for learning generative models, which can be expressed as a zero-sum game between two neural networks. The generator network produces samples from the arbitrary given distribution, while the adversarial discriminator tries to distinguish between real data and generated data. Meanwhile, the generator network tries to fool the discriminator network by producing plausible samples which are close to real samples. When a final theoretical equilibrium is achieved, discriminator can never distinguish between real and fake data. However, we show that a theoretical equilibrium often can not be achieved with discrete finite samples in datasets during the training process in practice.

Although GANs have achieved remarkable progress, numerous researchers have tried to improve the performance of GANs from various aspects (Arjovsky et al. (2017); Nowozin et al. (2016); Gulrajani et al. (2017); Miyato et al. (2018)) because of the inherent problem in GAN training, such as instability and mode collapse. Arora et al. (2017) showed that a theoretical generalization guarantee does not be provided with the original GAN objective and analyzed the generalization capacity of neural network distance. The author argued that for a low capacity discriminator, it can not provide generator enough information to fit the target distribution owing to lack of ability to detect mode collapse. Thanh-Tung et al. (2019) argued that poor generation capacity in GANs comes from discriminators trained on discrete finite datasets resulting in overfitting to real data samples and gradient exploding when generated datapoints approach real ones. As a result, Thanh-Tung et al. (2019) proposed a zero-centered gradient penalty on linear interpolations between real and fake samples (GAN-0GP-interpolation) to improve generalization capability and prevent mode collapse resulted from gradient exploding. Recent work Wu et al. (2019) further studied generalization from a new perspective of privacy protection.

In this paper, we focus on mode collapse resulted from gradient exploding studied in Thanh-Tung et al. (2019) and achieve a better generalization with a much more stable training process. Our contributions are as follows:
Table 1: NOTATIONS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_r$</td>
<td>the target distribution</td>
</tr>
<tr>
<td>$p_g$</td>
<td>the model distribution</td>
</tr>
<tr>
<td>$d_x$</td>
<td>the dimensionality of a data sample</td>
</tr>
<tr>
<td>$D$</td>
<td>discriminator with sigmoid function in the last layer</td>
</tr>
<tr>
<td>$D_0$</td>
<td>discriminator with sigmoid function in the last layer removed</td>
</tr>
<tr>
<td>$D_r = {x_1, \ldots, x_n}$</td>
<td>the set of $n$ real samples</td>
</tr>
<tr>
<td>$D_g = {y_1, \ldots, y_m}$</td>
<td>the set of $m$ generated samples</td>
</tr>
<tr>
<td>$D_f = {f_1, \ldots, f_m}$</td>
<td>the candidate set of $M_1$ generated samples to be selected as real</td>
</tr>
<tr>
<td>$D_{FAR} \subset {f_1, \ldots, f_m}$</td>
<td>the set of $M_0$ generated samples considered as real</td>
</tr>
</tbody>
</table>

1. We show that a theoretical equilibrium, when optimal discriminator outputs a constant for both real and generated data, is unachievable for an empirical discriminator during the training process. Due to this fact, it is possible that gradient exploding happens when fake datapoints approach real ones, resulting in an unbalanced generated distribution that deviates from the target one.

2. We show that when generated datapoints are very close to real ones in distance, penalizing the difference between discriminator outputs and considering fake as real can alleviate gradient exploding to prevent overfitting to certain real datapoints.

3. We show that when more fake datapoints are moved towards a single real datapoint, gradients of the generator on fake datapoints very close to the real one can not be reduced, which partly explains the reason of a more serious overfitting phenomenon and an increasingly unbalanced generated distribution.

4. Based on the zero-centered gradient penalty on data samples (GAN-0GP-sample) proposed in Mescheder et al. (2018), we propose a novel GAN training method by considering some fake samples as real ones according to the discriminator outputs in a training batch to effectively prevent mode collapse. Experiments on synthetic and real world datasets verify that our method can stabilize the training process and achieve a more faithful generated distribution.

In the sequel, we use the terminologies of generated samples (datapoints) and fake samples (data-points) indiscriminately. Tab. 1 lists some key notations used in the rest of the paper.

2 RELATED WORKS

Unstability. GANs have been considered difficult to train and often play an unstable role in training process [Salimans et al. (2016)]. Various methods have been proposed to improve the stability of training. A lot of works stabilized training with well-designed structures [Radford et al. (2015); Karras et al. (2018); Zhang et al. (2019); Chen et al. (2019)] and utilizing better objectives [Nowozin et al. (2016); Zhao et al. (2016); Arjovsky et al. (2017); Mao et al. (2017)]. Gradient penalty to enforce Lipschitz continuity is also a popular direction to improve the stability including Gulrajani et al. (2017); Petzka et al. (2018); Roth et al. (2017); O. (2017). From the theoretical aspect, Nagarajan & Kolter (2017) showed that GAN optimization based on gradient descent is locally stable and Mescheder et al. (2018) proved local convergence for simplified zero-centered gradient penalties under suitable assumptions. For a better convergence, a two time-scale update rule (TTUR) [Heusel et al. (2017)] and exponential moving averaging (EMA) [Yazici et al. (2019)] have also been studied.

Mode collapse. Mode collapse is another persistent essential problem for the training of GANs, which means lack of diversity in the generated samples. The generator may sometimes fool the discriminator by producing a very small set of high-probability samples from the data distribution. Recent work [Arora et al. (2017); Arora et al. (2018)] studied the generalization capacity of GANs and showed that the model distributions learned by GANs do miss a significant number of modes.
A large number of ideas have been proposed to prevent mode collapse. Multiple generators are applied in [Arora et al. (2017); Ghosh et al. (2018); Hoang et al. (2018)] to achieve a more faithful distribution. Mixed samples are considered as the inputs of discriminator in [Lin et al. (2018); Lucas et al. (2018)] to convey information on diversity. Recent work [He et al. (2019)] studied mode collapse from probabilistic treatment and [Yamaguchi & Koyama (2019)] from entropy of distribution.

3 BACKGROUND

In the original GAN [Goodfellow et al. (2014)], the discriminator $D$ maximizes the following objective:

$$
\mathcal{L} = \mathbb{E}_{x \sim p_r} [\log(D(x))] + \mathbb{E}_{y \sim p_g} [\log(1 - D(y))].
$$

(1)

The logistic sigmoid function $\sigma(x) = \frac{1}{1 + e^{-x}}$ is usually used in practice, leading to

$$
\mathcal{L} = \mathbb{E}_{x \sim p_r} [\log(\sigma(D_0(x)))] + \mathbb{E}_{y \sim p_g} [\log(1 - \sigma(D_0(y))]),
$$

(2)

and to prevent gradient collapse, the generator $G$ maximizes

$$
\mathcal{L}_G = \mathbb{E}_{y \sim p_g} [\log(\sigma(D_0(y)))]
$$

(3)

where $D_0$ is usually represented by a neural network. [Goodfellow et al. (2014)] showed that the optimal discriminator $D$ in Eqn. 1 is $D^*(v) = \frac{p_r(p)}{p_r(p) + p_g(v)}$ for any $v \in \text{supp}(p_r) \cup \text{supp}(p_g)$. As the training progresses, $p_g$ will be pushed closer to $p_r$. If $G$ and $D$ are given enough capacity, a global equilibrium is reached when $p_r = p_g$, in which case the best strategy for $D$ on $\text{supp}(p_r) \cup \text{supp}(p_g)$ is just to output $\frac{1}{2}$ and the optimal value for Eqn. 1 is $2 \log(\frac{1}{2})$.

With finite training examples in training dataset $D_r$ in practice, an empirical version is applied to approximate Eqn. 1 using $\frac{1}{n} \sum_{i=1}^n \log(D(x_i))$ to estimate $\mathbb{E}_{x \sim p_r} [\log(D(x))]$ and $\frac{1}{m} \sum_{i=1}^m [1 - \log(D(y_i))]$ to estimate $\mathbb{E}_{y \sim p_g} [\log(1 - D(y))]$, where $x_i$ is from the set $D_r$ of $n$ real samples and $y_i$ is from the set $D_g$ of $m$ generated samples, respectively.

Mode collapse in the generator is attributed to gradient exploding in discriminator, according to [Thanh-Tung et al. (2019)]. When a fake datapoint $y_0$ is pushed to a real datapoint $x_0$ and if $|D(x_0) - D(y_0)| \geq \epsilon$, it is satisfied, the absolute value of directional derivative of $D$ in the direction $\mu = x_0 - y_0$ will approach infinity leading to gradient exploding:

$$
|\nabla_\mu D(x_0)| = \lim_{y_0 \mapsto x_0} \frac{|D(x_0) - D(y_0)|}{||x_0 - y_0||} \geq \lim_{y_0 \mapsto x_0} \frac{\epsilon}{||x_0 - y_0||} = \infty.
$$

(4)

Since the gradient $\nabla_{y_0} D(y_0)$ at $y_0$ outweighs gradients towards other modes in a mini-batch, gradient exploding at datapoint $y_0$ will move multiple fake datapoints towards $x_0$ resulting in mode collapse.

4 OPTIMAL DISCRIMINATOR IN EMPIRICAL VERSION

4.1 OPTIMAL DISCRIMINATOR EMPIRICALLY DOES NOT HAVE A THEORETICAL EQUILIBRIUM

Theoretically discriminator outputs a constant $\frac{1}{2}$ when a global equilibrium is reached. However in practice, discriminator can often easily distinguish between real samples and fake samples ([Goodfellow et al. (2014); Arjovsky et al. (2017)]), making a theoretical equilibrium unachievable. Because the distribution $p_r$ of real data is unknown for discriminator, discriminator will always consider datapoints in the set $D_r$ of real samples as real while $D_g$ of generated samples as fake. Even when generated distribution $p_g$ is equivalent to the target distribution $p_r$, $D_r$ and $D_g$ is disjoint with probability 1 when they are sampled from two continuous distributions respectively (proposition 1 in [Thanh-Tung et al. (2019)]). In this case, actually $p_g$ is pushed towards samples in $D_r$ instead of the target distribution. However, we show next because of the fact of an unachievable theoretical equilibrium for empirical discriminator during the training process, an unbalanced distribution would be generated that deviates from the target distribution.
Let’s first consider a simplified scenario where a fake datapoint $y$ alleviates gradient exploding to achieve a more faithful generated distribution. The following subsections, we are interested in the way of stabilizing the output of discriminator to $D$, generated distribution neither covers the target Gaussian distribution nor fits all the real samples in $D_r$, collapse. See the generated results on a Gaussian dataset of original GAN in Fig. 1a and 1e. The and overfitting to a single datapoint happen, resulting an unbalanced distribution and visible mode collapse. See the generated results on a Gaussian dataset of original GAN in Fig. 1a and 1b. The generated distribution neither covers the target Gaussian distribution nor fits all the real samples in $D_r$, making an unbalanced distribution visible. Furthermore, in practice discriminator and generator are represented by a neural network with finite capacity and dataset $D_r$ is relatively huge, generator can never memorize every discrete sample resulting in a theoretical equilibrium unachievable. In the following subsections, we are interested in the way of stabilizing the output of discriminator to alleviate gradient exploding to achieve a more faithful generated distribution.

4.2 Penalizing the difference between discriminator outputs on close real and fake pairs

Let’s first consider a simplified scenario where a fake datapoint $y_0$ is close to a real datapoint $x_0$. Generator updates $y_0$ according to the gradient that the generator receives from the discriminator with respect to the fake datapoint $y_0$, which can be computed as:

$$\nabla_{y_0} \mathcal{L}_D(y_0) = \frac{\partial \log(\sigma(D_0(y_0)))}{\partial D_0(y_0)} \nabla_{y_0} D_0(y_0) = \frac{\nabla_{y_0} D_0(y_0)}{1 + e^{D_0(y_0)}}. \quad (5)$$

When $y_0$ approaches $x_0$ very closely and a theoretical discriminator equilibrium is not achieved here, namely $D_0(x_0) - D_0(y_0) \geq \epsilon$, the absolute value of directional derivative $(\nabla_{\mu} D_0)_{y_0}$ in the direction $\mu = x_0 - y_0$ at $y_0$ tends to explode and will outweigh directional derivatives in other

**Proposition 1.** For empirical discriminator in original GAN, unless $p_g$ is a discrete uniform distribution on $D_r$, the optimal discriminator $D$ output on $D_r$ and $D_g$ is not a constant $\frac{1}{2}$, since there exists a more optimal discriminator which can be constructed as a MLP with $O(2d_x)$ parameters.

See Appendix A for the detailed proof. If all the samples in $D_r$ can be remembered by discriminator and generator, and only if generated samples can cover $D_r$ uniformly, which means $D_g = D_r$, a theoretical equilibrium in discriminator can be achieved. However, before generator covers all the samples in $D_r$ uniformly during the training process, the fact of an unachievable theoretical equilibrium makes it possible that there exists a real datapoint $x_0$ with a higher discriminator output than that of a generated datapoint $y_0$. When $y_0$ approaches $x_0$ very closely, gradient exploding and overfitting to a single datapoint happen, resulting an unbalanced distribution and visible mode collapse. See the generated results on a Gaussian dataset of original GAN in Fig. 1a and 1b. The generated distribution neither covers the target Gaussian distribution nor fits all the real samples in $D_r$, making an unbalanced distribution visible. Furthermore, in practice discriminator and generator are represented by a neural network with finite capacity and dataset $D_r$ is relatively huge, generator can never memorize every discrete sample resulting in a theoretical equilibrium unachievable. In the following subsections, we are interested in the way of stabilizing the output of discriminator to alleviate gradient exploding to achieve a more faithful generated distribution.

Figure 1: Results on finite samples from a Gaussian distribution of GANs trained with different gradient penalties and our method. (blue datapoints represent real samples and red datapoints represent generated samples) (a),(c) GAN with no GP, iter. 100,000 and 200,000. (b),(f) GAN-0Gp-sample, iter. 100,000 and 200,000. (c),(g) GAN-0Gp-interpolation, iter. 100,000 and 200,000. (d),(h) GAN-0Gp-sample with our method, iter. 100,000 and 200,000.
Table 2: Changes of different optimal values with variables discussed in Section 4 (↗ means increasing and ↘ means decreasing)

<table>
<thead>
<tr>
<th>$k_0$</th>
<th>$m_0$</th>
<th>$p_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>↓</td>
<td>↓</td>
<td>↓</td>
</tr>
</tbody>
</table>

Based on Proposition 2, penalizing the difference between discriminator outputs on close real and fake datapoints. It comes naturally to penalize for real datapoints and that for generated ones. Thus, an extra constraint is needed to alleviate the overfitting to a single real one. Note that the output of $D_0$ and $D_0(x_0)$ is very close to a close real and fake pair. We are specially interested in the optimal discriminator output at $x_0$ and $D_0(y_0)$ as $ξ_0$ and $D_0(y_0)$ as $ξ_1$, $i = 1, \cdots, m_0$, we have the following empirical discriminator objective:

$$
\mathcal{L} = \frac{1}{n} \left( \log \sigma(ξ_0) + \sum_{i=2}^{n} \log \sigma(D_0(x_i)) \right) + \frac{1}{m} \left( \sum_{i=1}^{m_0} \log(1 - \sigma(ξ_i)) + \sum_{i=m_0+1}^{m} \log(1 - \sigma(D_0(y_i))) \right) - \frac{k_0}{m_0} \sum_{i=1}^{m_0} (ξ_0 - ξ_i)^2
$$

$$
= C_1 + \frac{1}{n} f(ξ_0, ξ_1, \cdots, ξ_{m_0}),
$$

where the interested term $f(ξ_0, ξ_1, \cdots, ξ_{m_0})$ is

$$
f(ξ_0, ξ_1, \cdots, ξ_{m_0}) = \log \sigma(ξ_0) + \frac{n}{m} \sum_{i=1}^{m_0} \log(1 - \sigma(ξ_i)) - \frac{k_0}{m_0} \sum_{i=1}^{m_0} (ξ_0 - ξ_i)^2.
$$

**Proposition 2.** Assume $\{ξ_0^*, \cdots, ξ_{m_0}^*\}$ is an optimizer that achieves the maximum value of $f(ξ_0, ξ_1, \cdots, ξ_{m_0})$. Then with $k_0$ increasing, $ξ_0^*$ decreases, $ξ_1^*$ increases and $ξ_0^* - ξ_1^*$ decreases, and, with $m_0$ increasing, $ξ_0^*$ decreases, $ξ_1^*$ decreases and $ξ_0^* - ξ_1^*$ increases, where $i = 1, \cdots, m_0$.

**Proof.** See Appendix [B] □

Based on Proposition 2, penalizing the difference between discriminator outputs on close real and fake pairs $\{x_0, y_i\}$ can reduce the norm of $\nabla_y \mathcal{L}_{\mathcal{G}}(y_i)$ from Eqn. 6 making it possible to move fake
datapoints to other real datapoints instead of only being trapped at \( x_0 \). However in practice, it is hard to find the close real and fake pairs to penalize the corresponding difference between discriminator outputs. If we directly penalize the \( L_2 \) norm of \( D_0(x_i) - D_0(y_i) \) when \( \{x_i, y_i\} \) are not a pair of close datapoints, \( ||\nabla y_i L_G(y_i)|| \) for \( y_i \) may even get larger. Consider \( D_0(y_i) \) has a higher value than \( D_0(x_i) \), which could happen when \( x_i \) has more corresponding close fake datapoints than the real datapoint \( x_u \) corresponding to \( y_i \) from Proposition 2. Direct penalization will make \( D_0(y_i) \) lower, then \( D_0(x_u) - D_0(y_i) \) gets higher and \( ||\nabla y_i L_G(y_i)|| \) higher. Thus in practice we could enforce a zero-centered gradient penalty of the form \( ||(\nabla D_0)_v||^2 \) to stabilize discriminator output, where \( v \) can be real datapoints or fake datapoints. Although Thanh-Tung et al. (2019) thought that discriminator can have zero gradient on the training dataset and may still have gradient exploding outside the training dataset, we believe a zero-centered gradient penalty can make it harder for discriminator to distinguish between real and fake datapoints and fill the gap between discriminator outputs on close real and fake pairs to prevent overfitting to some extent. Fig. 1b and 1e alleviate overfitting phenomenon compared with no gradient penalty in Fig. 1a and 1c.

Thanh-Tung et al. (2019) proposed another zero-centered gradient penalty of the form \( ||(\nabla D_0)_v||^2 \), where \( v \) is a linear interpolation between real and fake datapoints, to prevent gradient exploding. However, we consider it’s not a very efficient method to fill the gap between discriminator outputs on close real and fake pairs. To begin with, the results of direct linear interpolation between real and fake datapoints may not lie in \( \text{supp}(p_r) \cup \text{supp}(p_f) \). Although the author also considered the interpolation on latent codes, it needs an extra encoder which increases operational complexity. Furthermore, for arbitrary pair of real and fake datapoints, the probability that linear interpolation between them lie where gradient exploding happens is close to 0, because large gradient happens when a fake datapoint approaches closely a real datapoint, resulting in the gap between discriminator outputs on close real and fake pairs hard to fill.

Based on Proposition 2, we also find that when more fake datapoints are moved to the corresponding real datapoint, \( ||\nabla y_i L_G(y_i)|| \) for a fake datapoint \( y_i \) only to increase from Eqn. 6. It means with the training process going on, more fake datapoints tend to be attracted to one single real datapoint and it gets easier to attract much more fake datapoints to the real one. It partly explains the unstability of GAN training process that especially during the latter stage of training, similar generated samples are seen. Compared with Fig. 1a, 1b and 1c at iter.100,000, Fig. 1e, 1f and 1g at iter.200,000 have a worse generalization and much more similar samples are generated with the training process going on.

4.3 CONSIDERING FAKE AS REAL ON CLOSE REAL AND FAKE PAIRS

In this subsection, we aim to make \( ||\nabla y_i L_G(y_i)||, i = 1, \cdots, m_0 \) smaller for optimal empirical discriminator by considering some fake as real on close real and fake pairs based on the above discussions. Suppose for each fake datapoint, it’s considered as real datapoint with probability \( p_0 \) when training real datapoints, resulting in the following empirical discriminator objective:

\[
L = \frac{1}{n} \left( \log \sigma(\xi_0) + E[A \sum_{i=1}^{m_0} \log \sigma(\xi_i)] + \sum_{i=2}^{m} \log \sigma(D_0(x_i)) \right) \\
+ \frac{1}{m} \left( \sum_{i=1}^{m_0} \log(1 - \sigma(\xi_i)) + \sum_{i=m_0+1}^{m} \log(1 - \sigma(D_0(y_i))) \right) \\
- \frac{k_0}{m_0} \sum_{i=1}^{m_0} (\xi_0 - \xi_i)^2 \\
= C_2 + \frac{1}{n} h(\xi_0, \xi_1, \cdots, \xi_{m_0}), \quad (9)
\]

where \( A \) is a binary random variable taking values in \( \{0, 1\} \) with \( \text{Pr}(A = 1) = p_0 \) and the interested term \( h(\xi_0, \xi_1, \cdots, \xi_{m_0}) \) is

\[
h(\xi_0, \xi_1, \cdots, \xi_{m_0}) = \log \sigma(\xi_0) + p_0 \sum_{i=1}^{m_0} \log \sigma(\xi_i) + \frac{n}{m} \sum_{i=1}^{m_0} \log(1 - \sigma(\xi_i)) - \frac{n k_0}{m_0} \sum_{i=1}^{m_0} (\xi_0 - \xi_i)^2. \quad (10)
\]
Proposition 3. Assume \( \{ \xi_0, \cdots, \xi_{m_0} \} \) is an optimizer that achieves the maximum value of \( h(\xi_0, \xi_1, \cdots, \xi_{m_0}) \). Then with \( p_0 \) increasing, \( \xi_0 \) increases, \( \xi_i \) increases and \( \xi_0 - \xi_i \) decreases, where \( i = 1, \cdots, m_0 \).

Proof. See Appendix C

Note that only penalizing the difference between discriminator outputs on close real and fake pairs in Subsection 4.2 is just a special case of considering fake as real here when \( p_0 = 0 \). Based on Proposition 3 considering fake datapoints as real with increasing probability \( p_0 \) for real dataponts training part can reduce the norm of \( \nabla_p \) the overfitting to one single real datapoint. Note that for a fixed fake as real where large gradient happens for real training part, the attraction to the real datapoint for fake ones can be alleviate to make it easier to be moved to other real datapoints and prevent the overfitting to one single real datapoint. Note that for a fixed \( p_0 \), when the number \( m_0 \) of fake datapoints very close to the real one increases, more fake datapoints will be considered as real to alleviate the influences of increasing \( m_0 \) discussed in Subsection 4.2.

5 Method

To overcome the problem of overfitting to some single datapoints and achieve a better generalization, we propose that fake samples generated by generator in real time can be trained as real samples in discriminator. For original \( N \) real samples in a training batch in training process, we substitute them with \( N_0 \) real samples in \( D_r \) and \( M_0 \) generated samples in \( D_g \), where \( N = N_0 + M_0 \). Our approach is mainly aimed at preventing large gradient in the region where many generated samples overlap one single real sample. To find generated samples in these regions, we choose the generated ones with low discriminator output, owing to the reason that discriminator tends to have a lower output for the region with more generated datapoints approaching one real datapoint from Proposition 2. Therefore, we choose needed \( M_0 \) generated samples denoted as set \( D_{FAR} \) as real samples from a larger generated set \( D_f \) containing \( M_1 \) generated samples \( \{ f_1, f_2, \cdots, f_{M_1} \} \) according to corresponding discriminator output:

\[
D_{FAR} = \{ f_i \}, i \in \text{index of top } M_0 \text{ in } \{-D_0(f_1), -D_0(f_2), \cdots, -D_0(f_{M_1})\}. \tag{11}
\]

We also add a zero-centered gradient penalty on real datapoints [Mescheder et al., 2018] based on the discussions in Subsection 4.2, resulting in the following empirical discriminator objective in a batch containing \( N \) real samples and \( M \) fake samples:

\[
\mathcal{L}_{FAR} = \frac{1}{N} \sum_{i=1}^{N_0} \log(\sigma(D_0(x_i))) + \sum_{i=1}^{M_0} \log(\sigma(D_0(f_i))))] + \frac{1}{M} \sum_{i=1}^{M} \log(1 - \sigma(D_0(y_i))) + \frac{\lambda}{N} \sum_{i=1}^{N} ||(\nabla D_0)c_i||^2, \tag{12}
\]

where \( x_i \in D_r, f_i \in D_{FAR}, y_i \in D_g \) and \( c_i \in D_r \cup D_{FAR} \). In practice, we usually let \( N = M \). Because some fake datapoints are trained as real ones, the zero-centered gradient penalty are actually enforced on the mixture of real and fake datapoints.

When we sample more generated datapoints for \( D_f \) to decide the needed \( M_0 \) datapoints as real, the probability of finding the overfitting region with large gradient is higher. When more fake datapoints in \( D_{FAR} \) that are close to corresponding real ones are considered as real for training, it is equivalent to increase the value of \( p_0 \) in Subsection 4.3. For a larger \( D_{FAR} \), the number of real samples \( N_0 \) will decrease for a fixed batchsize \( N \) and the speed to cover real ones may be slowed at the beginning of training owning to the reason that some fake datapoints are considered as real and discriminator will not be so confident to give fake ones a large gradient to move them to real ones. Our method can stabilize discriminator output and prevent mode collapse caused by gradient exploding efficiently based on our theoretical analysis. A more faithful generated distribution will be achieved in practice.
6 Experiments

6.1 Synthetic data

To test the effectiveness of our method in preventing an unbalanced distribution resulted from overfitting to only some real datapoints, we designed a dataset with finite real samples coming from a Gaussian distributions and trained MLP based GANs with different gradient penalties and our method on that dataset. For gradient penalties in all GANs, the weight $\lambda$ is set 10. Training batch is set 64 and one quarter of the real training batch are generated samples picked from 256 generated samples according to discriminator output, namely $M_0 = 16$ and $M_1 = 256$ in Eqn. [1]. Learning rate is set 0.003 for both G and D. The result is shown in Fig[1]. It can be observed that original GAN, GAN-0GP-sample and GAN-0GP-interpolation all have serious overfitting problem leading to a biased generated distribution with training process going on, while our method can generate much better samples with good generalization.

We also test our method on a mixture of 8 Gaussians dataset where random samples in different modes are far from each other. The evolution of our method is depicted in Fig[2]. We observe that although our method only covers 3 modes at the beginning, it can cover other modes gradually because our method alleviates the gradient exploding on close real and fake datapoints. It is possible that fake datapoints are moved to other Gaussian modes when the attraction to other modes is larger than to the overfitted datapoints. Hence, our method has the ability to find the uncovered modes to achieve a faithful distribution even when samples in high dimensional space are far from each other. More synthetic experiments can be found in Appendix D.

Figure 2: Evolution of our method on a mixture of 8 Gaussians dataset. (a) iter. 0. (b) iter. 100,000. (c) iter. 335,000. (d) iter. 500,000.

6.2 Real world data

To test our method on real world data, we compare our method with GAN-0GP-sample on CIFAR-10 (Antonio et al. [2008]), CIFAR-100 (Antonio et al. [2008]) and a more challenging dataset ImageNet (Russakovsky et al. [2015]) with ResNet-architectures similar with that in Mescheder et al. [2018]. Inception score (Salimans et al. [2016]) and FID (Heusel et al. [2017]) are used as quantitative
measures. For Inception score, we follow the guideline from Salimans et al. (2016). The FID score is evaluated on 10k generated images and statistics of data are calculated at the same scale of generation. Better generation can be achieved with higher inception score and lower FID value. The maximum number of iterations for CIFAR experiment is 500k, while for ImageNet is 600k because of training difficulty with much more modes. We use the code from Mescheder et al. (2018).

The weight $\lambda$ for gradient penalty is also set 10. Training batch is set 64 and for a better gradient alleviation on close real and fake datapoints, half of the real training batch are generated samples with $M_0 = 32$ and $M_1 = 256$ in Eqn. [11]. For CIFAR experiments, we use the RMSProp optimizer with $\alpha = 0.99$ and a learning rate of $10^{-3}$. For ImageNet experiments, we use the Adam optimizer with $\alpha = 0$, $\beta = 0.9$ and TTUR with learning rates of $10^{-4}$ and $3 \times 10^{-4}$ for the generator and discriminator respectively. We use an exponential moving average with decay 0.999 over the weights to produce the final model.

6.2.1 CIFAR-10 AND CIFAR-100

The results on Inception score and FID are shown in Fig. 3 and 4. Our method outperforms GAN-0GP-sample by a large margin. As predicted in Section 5, the speed of our method to cover real ones could be slowed at the beginning of training with some fake considered as real. However, our method can cover more modes and have a much better balanced generation than the baseline.

The losses of discriminator and generator during the process of CIFAR-10 training are shown in Fig. 5. It can be observed that our method has a much more stable training process. Owing to the overfitting to some single datapoints and an unbalanced generated distribution missing modes, the losses of discriminator and generator for GAN-0GP-sample gradually deviate from the optimal theoretical value, namely $2 \log 2 \approx 1.386$ for discriminator and $\log 2 \approx 0.693$ for generator respectively. However, our method has a much more stable output of discriminator to achieve the losses for discriminator and generator very close to theoretical case. It proves practically that our method can stabilize discriminator output on close real and fake datapoints to prevent more datapoints from
trapped in a local region and has a better generalization. The losses of discriminator and generator on CIFAR-100 and image samples can be found in Appendix D.

6.2.2 IMAGE NET

For the challenging ImageNet task, we train GANs to learn a generative model of all 1000 classes at resolution $64 \times 64$ with the limitation of our hardware. However, our models are completely unsupervised learning models with no labels used instead of another 256 dimensions being used in latent code $z$ as labels in [Mescheder et al., 2018].

The results in Fig. 6 show that our methods still outperforms GAN-0GP-sample on ImageNet. Our method can produce samples of state of the art quality without using any category labels and stabilize the training process. Random selected samples and losses of discriminator and generator during the training process can be found in Appendix D.

7 CONCLUSION

In this paper, we explain the reason that an unbalanced distribution is often generated in GANs training. We show that a theoretical equilibrium for empirical discriminator is unachievable during the training process. We analyze the affection on the gradient that generator receives from discriminator with respect to restriction on difference between discriminator outputs on close real and fake pairs and trick of considering fake as real. Based on the theoretical analysis, we propose a novel GAN training method by considering some fake samples as real ones according to the discriminator outputs in a training batch. Experiments on diverse datasets verify that our method can stabilize the training process and improve the performance by a large margin.

REFERENCES


Shoichiro Yamaguchi and Masanori Koyama. DISTRIBUTIONAL CONCAVITY REGULARIZATION FOR GANS. In International Conference on Learning Representations, 2019.


A PROOF FOR PROPOSITION 1

For empirical discriminator, it maximizes the following objective:

\[ L = \mathbb{E}_{x \in D_r}[\log(D(x))] + \mathbb{E}_{y \in D_g}[\log(1 - D(y))] \]  

(13)

When \( p_g \) is a discrete uniform distribution on \( D_r \), and generated samples in \( D_g \) are the same with real samples in \( D_r \). It is obvious that the discriminator outputs \( \frac{1}{2} \) to achieve the optimal value when it cannot distinguish fake samples from real ones.

For continues distribution \( p_g \) [Thanh-Tung et al. (2019)] has proved that an \( \epsilon \)-optimal discriminator can be constructed as a one hidden layer MLP with \( O(d_x (m + n)) \) parameters, namely \( D(x) \geq \frac{1}{2} + \frac{\epsilon}{2}, \forall x \in D_r \) and \( D(y) \leq \frac{1}{2} - \frac{\epsilon}{2}, \forall y \in D_g \), where \( D_r \) and \( D_g \) are disjoint with probability 1. In this case, discriminator objective has a larger value than the theoretical optimal version:

\[ L \geq \mathbb{E}_{x \in D_r}[\log(\frac{1}{2} + \frac{\epsilon}{2})] + \mathbb{E}_{y \in D_g}[\log(\frac{1}{2} - \frac{\epsilon}{2})] \]

\[ = 2 \log\left(\frac{1}{2} + \frac{\epsilon}{2}\right) > 2 \log\frac{1}{2}. \]  

(14)

So the optimal discriminator output on \( D_r \) and \( D_g \) is not a constant \( \frac{1}{2} \) in this case.
Even discriminator has much less parameters than \( O(d_x(m + n)) \), there exists a real datapoint \( x_0 \) and a generated datapoint \( y_0 \) satisfying \( D(x_0) \geq \frac{1}{2} + \frac{\epsilon}{2} \) and \( D(y_0) \leq \frac{1}{2} - \frac{\epsilon}{2} \). Whether \( p_g \) is a discrete distribution only cover part samples in \( D_r \) or a continues distribution, there exists a generated datapoint \( y_0 \) satisfying \( y_0 \not\in D_r \). Assume that samples are normalized:

\[
\|x_i\| = \|y_i\| = 1, \forall x \in D_r, y \in D_g.
\]

Let \( W_1 \in \mathbb{R}^{2 \times d_x}, W_2 \in \mathbb{R}^{2 \times 2} \) and \( W_3 \in \mathbb{R}^2 \) be the weight matrices, \( b \in \mathbb{R}^2 \) offset vector and \( k_1, k_2 \) a constant, We can construct needed discriminator as a MLP with two hidden layer containing \( O(2d_x) \) parameters. We set weight matrices

\[
W_1 = \begin{bmatrix} x_0^T \\ y_0^T \end{bmatrix}, W_2 = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, W_3 = \begin{bmatrix} \frac{1}{\frac{1}{2} + \frac{\epsilon}{2}} \\ \frac{1}{\frac{1}{2} - \frac{\epsilon}{2}} \end{bmatrix}.
\]

For any input \( v \in D_r \cup D_g \), the discriminator output is computed as:

\[
D(v) = W_3^T \sigma(k_2 W_2 \sigma(k_1 (W_1 v - b)));
\]

where \( \sigma(x) = \frac{1}{1 + e^{-x}} \) is the sigmoid function. Let \( \alpha = W_1 v - b \), we have

\[
\alpha_1 = \begin{cases} 1 - b, & \text{if } v = x_0 \\ l - b, & \text{if } v \neq x_0 \end{cases}, \quad \alpha_2 = \begin{cases} 1 - b, & \text{if } v = y_0 \\ l - b, & \text{if } v \neq y_0 \end{cases};
\]

where \( l < 1 \). Let \( \beta = \sigma(k_1 \alpha) \), we have

\[
\beta_1 = \begin{cases} 1, & \text{if } v = x_0 \\ 0, & \text{if } v \neq x_0 \end{cases}, \quad \beta_2 = \begin{cases} 1, & \text{if } v = y_0 \\ 0, & \text{if } v \neq y_0 \end{cases};
\]

as \( k_1 \to \infty \) and \( b \to 1^- \). Let \( \gamma = \sigma(k_2 W_2 \beta) \), we have

\[
\gamma_1 = \begin{cases} 1, & \text{if } v = x_0 \\ 0, & \text{if } v = y_0 \end{cases}, \quad \gamma_2 = \begin{cases} 0, & \text{if } v = x_0 \\ 1, & \text{if } v = y_0 \end{cases};
\]

as \( k_2 \to \infty \). Hence, for any input \( v \in D_r \cup D_g \), discriminator outputs

\[
D(v) = W_3^T \gamma = \begin{cases} \frac{1}{2} + \frac{\epsilon}{2}, & \text{if } v = x_0 \\ \frac{1}{2} - \frac{\epsilon}{2}, & \text{if } v = y_0 \end{cases},
\]

In this case, discriminator objective also has a more optimal value than the theoretical optimal version:

\[
\mathcal{L} = \frac{1}{n} ((n - 1) \log \frac{1}{2} + \log \left( \frac{1}{2} + \frac{\epsilon}{2} \right)) + \frac{1}{m} ((m - 1) \log \frac{1}{2} + \log \left( \frac{1}{2} + \frac{\epsilon}{2} \right)) > 2 \log \frac{1}{2}.
\]

So the optimal discriminator output on \( D_r \) and \( D_g \) is also not a constant \( \frac{1}{2} \) in this case.

### B Proof for Proposition 2

We rewrite \( f(\xi_0, \xi_1, \cdots, \xi_{m_0}) \) here

\[
f(\xi_0, \xi_1, \cdots, \xi_{m_0}) = \log \sigma(\xi_0) + \frac{n}{m} \sum_{i=1}^{m_0} \log(1 - \sigma(\xi_i)) - \frac{nk_0}{m_0} \sum_{i=1}^{m_0} (\xi_0 - \xi_i)^2.
\]
To achieve the optimal value, let $f'(\xi_i) = 0, i = 0, \cdots, m_0$ and we have

$$
\begin{align*}
  f'(\xi_0) &= 1 - \sigma(\xi_0) - \frac{2nk_0}{m_0} \sum_{i=1}^{m_0} (\xi_0 - \xi_i) = 0, \quad (24) \\
  f'(\xi_i) &= -\frac{n}{m} \sigma(\xi_i) + \frac{2nk_0}{m_0} (\xi_0 - \xi_i) = 0, \quad i = 1, \cdots, m_0. \quad (25)
\end{align*}
$$

It is obvious that $\xi_1 = \xi_2 = \cdots = \xi_{m_0} = \xi$. Hence we have

$$
\begin{align*}
  1 - \sigma(\xi_0) - 2nk_0(\xi_0 - \xi) &= 0, \quad (26) \\
  -\frac{n}{m} \sigma(\xi) + \frac{2nk_0}{m_0} (\xi_0 - \xi) &= 0. \quad (27)
\end{align*}
$$

We can solve

$$
\xi = -\ln \left( \frac{nm_0}{m} (1 + e^{\xi_0}) - 1 \right). \quad (28)
$$

Substitute Eqn. (28) into Eqn. (26) and we get

$$
\begin{align*}
  f'(\xi_0) &= \frac{1}{1 + e^{\xi_0}} - 2nk_0(\xi_0 + \ln \left( \frac{nm_0}{m} (1 + e^{\xi_0}) - 1 \right)) = 0. \quad (29)
\end{align*}
$$

We can also have from Eqn. (28) and Eqn. (26) respectively

$$
\begin{align*}
  \xi_0 - \xi &= \xi_0 + \ln \left( \frac{nm_0}{m} (1 + e^{\xi_0}) - 1 \right), \quad (30) \\
  &= \frac{1}{2nk_0(1 + e^{\xi_0})}. \quad (31)
\end{align*}
$$

Note that there must exist an optimal $\xi_0$ satisfying $f'(\xi_0) = 0$ in Eqn. (29) so $\xi_0 + \ln \left( \frac{nm_0}{m} (1 + e^{\xi_0}) - 1 \right) > 0$ and $f'(\xi_0)$ in Eqn. (29) decreases with $k_0$ increasing. Also considering that $f'(\xi_0)$ is a monotonically decreasing function on $\xi_0$, $\xi_0$ decreases with $k_0$ increasing. From Eqn. (28) and Eqn. (30) we know $\xi^*$ increases and $\xi^*_0 - \xi^*$ decreases with $k_0$ increasing. Similarly, note that $f'(\xi_0)$ in Eqn. (29) decreases with $m_0$ increasing and $f'(\xi_0)$ is a monotonically decreasing function on $\xi_0$, we have that $\xi_0^*$ decreases with $m_0$ increasing. From Eqn. (31) we further know that $\xi^*$ decreases and $\xi^*_0 - \xi^*$ increases with $m_0$ increasing.

C PROOF FOR PROPOSITION 5

We rewrite $h(\xi_0, \xi_1, \cdots, \xi_{m_0})$ here

$$
\begin{align*}
  h(\xi_0, \xi_1, \cdots, \xi_{m_0}) &= \log \sigma(\xi_0) + p_0 \sum_{i=1}^{m_0} \log \sigma(\xi_i) + \frac{n}{m} \sum_{i=1}^{m_0} \log(1 - \sigma(\xi_i)) - \frac{nk_0}{m_0} \sum_{i=1}^{m_0} (\xi_0 - \xi_i)^2. \quad (32)
\end{align*}
$$

Let $f'(\xi_i) = 0, i = 0, \cdots, m_0$ and we have $\xi_1 = \xi_2 = \cdots = \xi_{m_0} = \xi$,

$$
\begin{align*}
  1 - \sigma(\xi_0) - 2nk_0(\xi_0 - \xi) &= 0, \quad (33) \\
  p_0(1 - \sigma(\xi_0)) - \frac{n}{m} \sigma(\xi) + \frac{2nk_0}{m_0} (\xi_0 - \xi) &= 0. \quad (34)
\end{align*}
$$

We can solve

$$
\xi = -\ln \left( \frac{nm_0}{m} (1 + e^{\xi_0}) - 1 \right). \quad (35)
$$

Substitute Eqn. (35) into Eqn. (33) and we get

$$
p_0 = \frac{1}{1 + e^{\xi_0}} \left[ e^{2nk_0\xi_0} - \frac{1}{1 + e^{\xi_0}} \left( \frac{nm_0}{m} (1 + e^{\xi_0}) - 1 \right)^2 - 2nk_0 \right] = g(\xi_0). \quad (36)
$$

The derivative of $g(\xi_0)$ with respect to $\xi_0$ is computed as

$$
\begin{align*}
  (1 + e^{\xi_0})^2 g'(\xi_0) &= e^{2nk_0\xi_0} - \frac{1}{1 + e^{\xi_0}} \left( 2nk_0 + \frac{e^{\xi_0}}{(1 + e^{\xi_0})^2} \right) \left( \frac{nm_0}{m} (1 + e^{\xi_0}) - 1 \right)^2 - 2nk_0 \\
  &= e^{2nk_0\xi_0} - \frac{1}{1 + e^{\xi_0}} \left( \frac{nm_0}{m} (1 + e^{\xi_0}) - 1 \right)^2 - 2nk_0 - \frac{nm_0}{m} e^{\xi_0} \left( 1 + e^{\xi_0} \right) - 2nk_0 \\
  &\quad + \left[ e^{2nk_0\xi_0} - \frac{1}{1 + e^{\xi_0}} \left( \frac{nm_0}{m} (1 + e^{\xi_0}) - 1 \right)^2 - 2nk_0 \right] \frac{e^{\xi_0}}{1 + e^{\xi_0}}. \quad (37)
\end{align*}
$$
Because
\[ \frac{nm_0}{m} (1 + e^{\xi_0}) - 1 = e^{-\xi} + (1 + e^{\xi_0})p_0 e^{-\xi} > 0 \] (38)
and
\[ \frac{1}{1 + e^{\xi_0}} e^{\xi_0} - \frac{1}{1 + e^{\xi_0}} \left( \frac{nm_0}{m} (1 + e^{\xi_0}) - 1 \right) \right] = p_0 \geq 0, \] (39)
g'\xi_0 > 0. Hence \( \xi_0^* \) increases with \( p_0 \) increasing. From Eqn. 33, we also have
\[ \xi_0 - \xi = \frac{1}{2n k_0 (1 + e^{\xi_0})}. \] (40)
we further know that \( \xi^* \) increases and \( \xi_0^* - \xi^* \) decreases with \( p_0 \) increasing.

D FURTHER RESULTS

Figure 7: Losses of discriminator (not including regularization term) and generator on CIFAR-100 of GAN-0GP-sample and GAN-0GP-sample with our method

Figure 8: Losses of discriminator (not including regularization term) and generator on ImageNet of GAN-0GP-sample and GAN-0GP-sample with our method

Figure 9: Generation of our method on a mixture of 25 Gaussians dataset and swissroll dataset.
Figure 10: Image generation of CIFAR-10.

Figure 11: Image generation of CIFAR-100.
Figure 12: Image generation of ImageNet.
E NETWORK ARCHITECTURES

For synthetic experiment, the network architectures are the same with that in Thanh-Tung et al. (2019). While for real world data experiment, we use the similar architectures in Mescheder et al. (2018). We use Pytorch (Paszke et al. (2017)) for development.

Table 3: Generator architecture in synthetic experiment

<table>
<thead>
<tr>
<th>Layer</th>
<th>output size</th>
<th>filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fully connected</td>
<td>64</td>
<td>2 → 64</td>
</tr>
<tr>
<td>RELU</td>
<td>64</td>
<td>-</td>
</tr>
<tr>
<td>Fully connected</td>
<td>64</td>
<td>64 → 64</td>
</tr>
<tr>
<td>RELU</td>
<td>64</td>
<td>-</td>
</tr>
<tr>
<td>Fully connected</td>
<td>64</td>
<td>64 → 64</td>
</tr>
<tr>
<td>RELU</td>
<td>64</td>
<td>-</td>
</tr>
<tr>
<td>Fully connected</td>
<td>2</td>
<td>64 → 2</td>
</tr>
</tbody>
</table>

Table 4: Discriminator architecture in synthetic experiment

<table>
<thead>
<tr>
<th>Layer</th>
<th>output size</th>
<th>filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fully connected</td>
<td>64</td>
<td>2 → 64</td>
</tr>
<tr>
<td>RELU</td>
<td>64</td>
<td>-</td>
</tr>
<tr>
<td>Fully connected</td>
<td>64</td>
<td>64 → 64</td>
</tr>
<tr>
<td>RELU</td>
<td>64</td>
<td>-</td>
</tr>
<tr>
<td>Fully connected</td>
<td>64</td>
<td>64 → 64</td>
</tr>
<tr>
<td>RELU</td>
<td>64</td>
<td>-</td>
</tr>
<tr>
<td>Fully connected</td>
<td>1</td>
<td>64 → 1</td>
</tr>
</tbody>
</table>

Table 5: Generator architecture in CIFAR experiment

<table>
<thead>
<tr>
<th>Layer</th>
<th>output size</th>
<th>filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fully connected</td>
<td>512 · 4 · 4</td>
<td>128 → 512 · 4 · 4</td>
</tr>
<tr>
<td>Reshape</td>
<td>512 × 4 × 4</td>
<td>-</td>
</tr>
<tr>
<td>Resnet-Block</td>
<td>256 × 4 × 4</td>
<td>512 → 256 → 256</td>
</tr>
<tr>
<td>NN-Upsampling</td>
<td>256 × 8 × 8</td>
<td>-</td>
</tr>
<tr>
<td>Resnet-Block</td>
<td>128 × 8 × 8</td>
<td>256 → 128 → 128</td>
</tr>
<tr>
<td>NN-Upsampling</td>
<td>128 × 16 × 16</td>
<td>-</td>
</tr>
<tr>
<td>Resnet-Block</td>
<td>64 × 16 × 16</td>
<td>128 → 64 → 64</td>
</tr>
<tr>
<td>NN-Upsampling</td>
<td>64 × 32 × 32</td>
<td>-</td>
</tr>
<tr>
<td>Resnet-Block</td>
<td>64 × 32 × 32</td>
<td>64 → 64 → 64</td>
</tr>
<tr>
<td>Conv2D</td>
<td>3 × 32 × 32</td>
<td>64 → 3</td>
</tr>
</tbody>
</table>
Table 6: Discriminator architecture in CIFAR experiment

<table>
<thead>
<tr>
<th>Layer</th>
<th>output size</th>
<th>filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv2D</td>
<td>64 x 32 x 32</td>
<td>3 → 64</td>
</tr>
<tr>
<td>Resnet-Block</td>
<td>128 x 32 x 32</td>
<td>64 → 64 → 128</td>
</tr>
<tr>
<td>Avh-Pool2D</td>
<td>128 x 16 x 16</td>
<td>-</td>
</tr>
<tr>
<td>Resnet-Block</td>
<td>256 x 16 x 16</td>
<td>128 → 128 → 256</td>
</tr>
<tr>
<td>Avh-Pool2D</td>
<td>256 x 8 x 8</td>
<td>-</td>
</tr>
<tr>
<td>Resnet-Block</td>
<td>512 x 8 x 8</td>
<td>256 → 256 → 512</td>
</tr>
<tr>
<td>Avh-Pool2D</td>
<td>512 x 4 x 4</td>
<td>-</td>
</tr>
<tr>
<td>Reshape</td>
<td>512 x 4 x 4</td>
<td>-</td>
</tr>
<tr>
<td>Fully Connected</td>
<td>1</td>
<td>512 x 4 x 4 → 1</td>
</tr>
</tbody>
</table>

Table 7: Generator architecture in ImageNet experiment

<table>
<thead>
<tr>
<th>Layer</th>
<th>output size</th>
<th>filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fully connected</td>
<td>1024 x 4 x 4</td>
<td>256 → 1024 x 4 x 4</td>
</tr>
<tr>
<td>Reshape</td>
<td>1024 x 4 x 4</td>
<td>-</td>
</tr>
<tr>
<td>Resnet-Block</td>
<td>1024 x 4 x 4</td>
<td>1024 → 1024 → 1024</td>
</tr>
<tr>
<td>NN-Upsampling</td>
<td>1024 x 8 x 8</td>
<td>-</td>
</tr>
<tr>
<td>Resnet-Block</td>
<td>512 x 8 x 8</td>
<td>1024 → 512 → 512</td>
</tr>
<tr>
<td>Reshape</td>
<td>512 x 8 x 8</td>
<td>-</td>
</tr>
<tr>
<td>Resnet-Block</td>
<td>256 x 8 x 8</td>
<td>128 → 256 → 256</td>
</tr>
<tr>
<td>Resnet-Block</td>
<td>256 x 16 x 16</td>
<td>256 → 256 → 256</td>
</tr>
<tr>
<td>NN-Upsampling</td>
<td>256 x 32 x 32</td>
<td>-</td>
</tr>
<tr>
<td>Resnet-Block</td>
<td>128 x 32 x 32</td>
<td>256 → 128 → 128</td>
</tr>
<tr>
<td>Resnet-Block</td>
<td>128 x 32 x 32</td>
<td>128 → 128 → 128</td>
</tr>
<tr>
<td>NN-Upsampling</td>
<td>128 x 64 x 64</td>
<td>-</td>
</tr>
<tr>
<td>Resnet-Block</td>
<td>64 x 64 x 64</td>
<td>128 → 64 → 64</td>
</tr>
<tr>
<td>Resnet-Block</td>
<td>64 x 64 x 64</td>
<td>64 → 64 → 64</td>
</tr>
<tr>
<td>Conv2D</td>
<td>3 x 64 x 64</td>
<td>64 → 3</td>
</tr>
</tbody>
</table>

Table 8: Discriminator architecture in ImageNet experiment

<table>
<thead>
<tr>
<th>Layer</th>
<th>output size</th>
<th>filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv2D</td>
<td>64 x 64 x 64</td>
<td>3 → 64</td>
</tr>
<tr>
<td>Resnet-Block</td>
<td>64 x 64 x 64</td>
<td>64 → 64 → 64</td>
</tr>
<tr>
<td>Resnet-Block</td>
<td>128 x 64 x 64</td>
<td>64 → 64 → 128</td>
</tr>
<tr>
<td>Avh-Pool2D</td>
<td>128 x 32 x 32</td>
<td>-</td>
</tr>
<tr>
<td>Resnet-Block</td>
<td>128 x 32 x 32</td>
<td>128 → 128 → 128</td>
</tr>
<tr>
<td>Avh-Pool2D</td>
<td>256 x 32 x 32</td>
<td>-</td>
</tr>
<tr>
<td>Resnet-Block</td>
<td>256 x 16 x 16</td>
<td>256 → 256 → 256</td>
</tr>
<tr>
<td>Avh-Pool2D</td>
<td>256 x 8 x 8</td>
<td>-</td>
</tr>
<tr>
<td>Resnet-Block</td>
<td>512 x 8 x 8</td>
<td>512 → 512 → 512</td>
</tr>
<tr>
<td>Avh-Pool2D</td>
<td>1024 x 8 x 8</td>
<td>-</td>
</tr>
<tr>
<td>Resnet-Block</td>
<td>1024 x 8 x 8</td>
<td>1024 → 1024 → 1024</td>
</tr>
<tr>
<td>Avh-Pool2D</td>
<td>1024 x 4 x 4</td>
<td>-</td>
</tr>
<tr>
<td>Resnet-Block</td>
<td>1024 x 4 x 4</td>
<td>1024 → 1024 → 1024</td>
</tr>
<tr>
<td>Fully Connected</td>
<td>1</td>
<td>1024 x 4 x 4 → 1</td>
</tr>
</tbody>
</table>