Improving GAN Training with Probability Ratio Clipping and Sample Reweighting

Yue Wu\(^1\), Pan Zhou\(^3\), Andrew Gordon Wilson\(^4\), Eric P. Xing\(^1,2\), Zhiting Hu\(^{1,2}\)
\(^1\)Carnegie Mellon University, \(^2\)Petuum Inc., \(^3\)National University of Singapore, \(^4\)New York University

Abstract

Despite success on a wide range of problems related to vision, generative adversarial networks (GANs) can suffer from inferior performance due to unstable training, especially for text generation. We propose a new variational GAN training framework which enjoys superior training stability. Our approach is inspired by a connection of GANs and reinforcement learning under a variational perspective. The connection leads to (1) probability ratio clipping that regularizes generator training to prevent excessively large updates, and (2) a sample re-weighting mechanism that stabilizes discriminator training by downplaying bad-quality fake samples. We provide theoretical analysis on the convergence of our approach. By plugging the training approach in diverse state-of-the-art GAN architectures, we obtain significantly improved performance over a range of tasks, including text generation, text style transfer, and image generation\(^1\).

1 Introduction

Generative adversarial networks (GANs) \(^{16}\) have drawn great research interests and achieved remarkable success in image synthesis \(^{4,40}\), video generation \(^{33}\), and others. However, it is usually hard to train a GAN well, because the training process is commonly unstable, subject to disturbances and even collapses. To alleviate this issue, substantial efforts have been paid to improve the training stability from different perspectives, e.g., divergence minimization \(^{37,38}\), Wasserstein distance with Lipschitz continuity of the discriminator \(^{2,18,51}\), energy-based models \(^{3,56}\), etc.

In spite of the above progresses, the instability in training has not been well resolved \(^9\), since it is difficult to well balance the strength of the generator and the discriminator. What is worse, such an instability issue is exacerbated in text generation due to the sequential and discrete nature of text \(^{6,14,36}\). Specifically, the high sensitivity of text generation to noise and the underlying errors caused by sparse discriminator signals in the generated text can often result in destructive updates to both generator and discriminator, enlarging the instability in GANs.

In this work, we develop a novel variational GAN training framework to improve the training stability, which is broadly applicable to GANs of a variety of architectures for image and text generation. This training framework is derived from a variational perspective of GANs and the resulting connections to reinforcement learning (in particular, RL-as-inference) \(^{11,50}\) and other rich literature \(^{5,17,25}\). Specifically, our approach consists of two stabilization techniques, namely, probability ratio clipping and sample re-weighting, for stabilizing the generator and discriminator respectively. (1) Under the new variational perspective, the generator update is subject to a KL penalty on the change of the generator distribution. This KL penalty closely resembles that in the popular Trust-Region Policy Optimization (TRPO) \(^{44}\) and its related Proximal Policy Optimization (PPO) \(^{45}\). This connection motivates a simple surrogate objective with a clipped probability ratio between the new generator and the old one. The probability ratio clipping discourages excessively large generator updates, and has

\(^1\)Code available at: github.com/Holmeswww/PPOGAN

shown to be effective in the context of stabilizing policy optimization \cite{45}. Figure 1 (left) shows the intuition about the surrogate objective, where we can observe the objective decreases with an overly large generator change and thus imposes regularization on the updates.

(2) When updating the discriminator, the new perspective induces an importance sampling mechanism, which effectively re-weights fake samples by their discriminator scores. Since low-quality samples tend to receive smaller weights, the discriminator trained on the re-weighted samples is more likely to maintain stable performance, and in turn provide informative gradients for subsequent generator updates. Figure 1 (middle/right) demonstrates the effect of the re-weighting in reducing the variance of both discriminator and generator losses. Similar importance weighting methods have recently been used in other contexts, such as de-biasing generative models \cite{17} and sampling from energy-based models \cite{12}. Our derivations can be seen as a variant for the new application of improving GANs.

We give theoretical convergence analysis showing the generator receives informative gradient \cite{57} from the generator. Empirically, we conduct extensive experiments on a range of tasks, including text generation, text style transfer, and image generation. Our approach shows significant improvement over state-of-the-art methods, demonstrating its broad applicability and efficacy.

\section{Related Work}

\textbf{Wasserstein distance, WGAN, and Lipschitz continuity.} The GAN framework \cite{16} features two components: a generator $G_\theta$ that synthesizes samples $x$ given some noise source $z$, namely $x = G_\theta(z)$ with $z \sim p_z(z)$, and a discriminator that distinguishes generator’s output and real data, which provides gradient feedback to improve the generator’s performance. WGAN \cite{2} improves the training stability of GANs by minimizing the Wasserstein distance $W(p_r, p_\theta)$ between the generation distribution $p_\theta$ (induced from $G_\theta$) and the real data distribution $p_r$. Its training loss is formulated as:

$$\min_\theta \max_{f \in \mathcal{F}} \mathbb{E}_{x \sim p_r}[f(x)] - \mathbb{E}_{x \sim p_\theta}[f(x)],$$

where $\mathcal{F}$ is the set of 1-Lipschitz functions; $f$ acts as the discriminator and is usually implemented by a neural network $f_\phi$. The original resort to enforce the Lipschitz constraint is through weight clipping \cite{2}. WGAN-GP \cite{18} later improves it by replacing it with a gradient penalty on the discriminator. CT-GAN \cite{51} further imposes the Lipschitz continuity constraint on the manifold of the real data $x \sim p_r$. Our approach is orthogonal to these prior works and can serve as a drop-in replacement to stabilize generator and discriminator in various kinds of GANs, such as WGAN-GP and CT-GAN.

Research on the Lipschitz continuity of GAN discriminators have resulted in the theory of “informative gradients” \cite{57} \cite{58}. Under certain mild conditions, a Lipschitz discriminator can provide informative gradient to the generator in a GAN framework: when $p_\theta$ and $p_r$ are disjoint, the gradient $\nabla f^*(x)$ of optimal discriminator $f^*$ w.r.t each sample $x \sim p_\theta$ points to a sample $x^* \sim p_r$, which guarantees that the generation distribution $p_\theta$ is moving towards $p_r$. We extend the informative gradient theory to our new case and show convergence of our approach.
3 Improving GAN Training

3.1 Motivations

Our approach is motivated by connecting GAN training with the well-established RL-as-inference methods \([1, 30]\) under a variational perspective. The connections enable us to augment GAN training with existing powerful probabilistic inference tools as well as draw inspirations from the rich RL literature for stable training. In particular, the connection to the popular TRPO \([44]\) and PPO \([45]\) yields the probability ratio clipping in generator training that avoids destructive updates (Sec. 3.2), and the application of importance sampling estimation gives rise to sample re-weighting for adaptive discriminator updates (Sec. 3.3). The full training procedure is then summarized in Alg. 1.

Specifically, as described in Sec. 2, the conventional formulation e.g., WGAN \([2]\) for updating the generator \(p_\theta(x)\) maximizes the expected discriminator score: \(\mathbb{E}_{p_\theta}[f_\phi(x)]\), where \(f_\phi\) is the Lipschitz-continuous discriminator parameterized with \(\phi\). The objective straightforwardly relates to policy optimization in RL by seeing \(p_\theta\) as a policy and \(f_\phi\) as a reward function. Thus, inspired by the probabilistic inference formulations of policy optimization \([1, 13, 25]\), here we transform the conventional objective by introducing a non-parametric auxiliary distribution \(q(x)\) and defining a new variational objective:

\[
\mathcal{L}(\theta, q) = \mathbb{E}_q[f_\phi(x)] - \text{KL}(q(x)\|p_\theta(x)),
\]

where KL is the KL divergence. Intuitively, we are maximizing the expected discriminator score of the auxiliary \(q\) (instead of generator \(p_\theta\)), and meanwhile encouraging the generator to stay close to \(q\).

As we shall see in more details shortly, the new formulation allows us to take advantage of off-the-shelf inference methods, which naturally leads to new components to improve the GAN training. In particular, maximizing the above objective is solved by the expectation maximization (EM) algorithm \([35]\) which alternatingly optimizes \(q\) at E-step and optimizes \(\theta\) at M-step. More specifically, at each iteration \(t\), given the current status of \(\theta = \theta^{(t)}\), the E-step that maximizes \(\mathcal{L}(\theta^{(t)}, q)\) w.r.t \(q\) has a closed-form solution:

\[
q^{(t)}(x) = \frac{p_{\theta^{(t)}}(x) \exp\{f_\phi(x)\}}{Z_\phi},
\]

where \(Z_\phi = \int p_{\theta^{(t)}}(x) \exp\{f_\phi(x)\} dx\) is the normalization term depending on the discriminator parameters \(\phi\). We elaborate on the M-step in the following, where we continue to develop the practical procedures for updating the generator and the discriminator, respectively.

3.2 Generator Training with Probability Ratio Clipping

The M-step optimizes \(\mathcal{L}(\theta, q^{(t)})\) w.r.t \(\theta\), which is equivalent to minimizing the KL divergence term in Eq. (2). However, since the generator \(p_\theta\) in GANs is often an implicit distribution that does not permit evaluating likelihood, the above KL term (which involves evaluating the likelihood of samples from \(q\)) is not applicable. We adopt an approximation, which has also been used in the classical wake-sleep algorithm \([21]\) and recent work \([25]\), by minimizing the reverse KL divergence as below. With Eq. (3) plugged in, we have:

\[
\min_\theta \text{KL}(p_\theta(x)\|q^{(t)}(x)) = \min_\theta -\mathbb{E}_{p_\theta}[f_\phi(x)] + \text{KL}(p_\theta(x)\|p_{\theta^{(t)}}(x)).
\]
As proven in the appendix, approximating with the reverse KL does not change the optimization problem. The first term on the right-hand side of the equation recovers the conventional objective of updating the generator. Of particular interest is the second term, which is a new KL regularizer between the generator \( p_\theta \) and its “old” state \( p_{\theta(t)} \) from the previous iteration. The regularizer discourages the generator from changing too much between updates, which is useful to stabilize the stochastic optimization procedure. The regularization closely resembles to that of TRPO/PPO, where a similar KL regularizer is imposed to prevent uncontrolled policy updates and make policy gradient robust to noises. Sec. [3.4] gives convergence analysis on the KL-regularized generator updates.

In practice, directly optimizing with the KL regularizer can be infeasible due to the same difficulty with the implicit distribution as above. Fortunately, PPO [45] has presented a simplified solution that emulates the regularized updates using a clipped surrogate objective, which is widely-used in RL. We adapt the solution to our context, leading to the following practical procedure of generator updates.

**Probability Ratio Clipping.** Let \( r_t \) denote the probability ratio \( r_t(\theta) = \frac{p_\theta(x)}{p_{\theta(t)}(x)} \) which measures the difference between the new and old distributions. For instance, \( r_t(\theta^{(t)}) = 1 \). The clipped surrogate objective for updating the generator, as adapted from PPO, is:

\[
L^{CLIP}(\theta) = \mathbb{E}_{p_\theta} \left[ \min\left( r_t(\theta) f_\phi(x), r_t^{\text{clip}}(\theta) f_\phi(x) \right) \right],
\]

where \( r_t^{\text{clip}}(\theta) = \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \) clips the probability ratio, so that moving \( r_t(\theta) \) outside of the interval \([1 - \epsilon, 1 + \epsilon]\) is discouraged. Taking the minimum puts a ceiling on the increase of the objective. Thus the generator does not benefit by going far away from the old generator.

Finally, to estimate the probability ratio \( r_t(\theta) \) when \( p_\theta \) is implicit, we use an efficient approximation similar to [7][17] by introducing a binary classifier \( C \) trained to distinguish real and generated samples. Assuming an optimal classifier \( C \) [7][16] which has \( p_\theta(x) = \frac{1 - C(x)}{C(x)} p_t(x) \), we then approximate \( r_t \) by keeping a historical version \( C \) of \( C^{(t)} \) at time-step \( t \):

\[
r_t(\theta) = \frac{p_\theta(x)}{p_{\theta(t)}(x)} \approx \frac{(1 - C(x)) \cdot C^{(t)}(x)}{(1 - C^{(t)}(x)) \cdot C(x)}. \tag{6}
\]

Note that after plugging the rightmost expression into Eq. (5), gradient can propagate through \( C \) to \( \theta \) since \( x = G_\theta(z) \). In practice, during the phase of generator training, we maintain \( C \) by fine-tuning it for one iteration every time after \( \theta \) is updated (Alg. [1]). Thus the maintenance of \( C \) is cheap. We give more details of \( C \) in the appendix.

### 3.3 Discriminator Training with Sample Re-weighting

We next discuss the training of the discriminator \( f_\phi \), where we augment the conventional training with an importance weighting mechanism for adaptive updates. Concretely, given the form of the auxiliary distribution solution \( q^{(t)} \) in Eq. (4), we first draw from the recent energy-based modeling work [8][16][25][27] and propose to maximize the data log-likelihood of \( q^{(t)} \) w.r.t \( \phi \). By taking the gradient, we derive the loss \( (\mathcal{L}(\phi)) \) for the discriminator \( f_\phi \):

\[
\nabla_\phi \mathcal{L}(\phi) = \nabla_\phi \left( \mathbb{E}_{p_t} \left[ f_\phi(x) \right] - \log Z_\phi \right) = \mathbb{E}_{p_t} \left[ \nabla_\phi f_\phi(x) \right] - \mathbb{E}_{q^{(t)}} \left[ \nabla_\phi f_\phi(x) \right]. \tag{7}
\]

We can observe that the resulting form resembles the conventional one (Sec. 2) as we are essentially maximizing \( f_\phi \) on real data while minimizing \( f_\phi \) on fake samples. An important difference is that here fake samples are drawn from the auxiliary distribution \( q^{(t)} \) instead of the generator \( p_\theta \). This difference leads to the new sample re-weighting component as below. Note that, as in WGAN (Sec. 2), we maintain \( f_\phi \) to be from the class of \( 1 \)-Lipschitz functions, which is necessary for the convergence analysis in Sec. [3.4]. In practice, we can use gradient penalty following [13][51].

**Sample Re-weighting.** We use the tool of importance sampling to estimate the expectation under \( q^{(t)} \) in Eq. (7). Given the multiplicative form of \( q^{(t)} \) in Eq. (3), similar to [1][12][25], we use the generator \( p_{\theta(t)} \) as the proposal distribution. This leads to

\[
\mathbb{E}_{q^{(t)}} \left[ \nabla_\phi f_\phi(x) \right] = \mathbb{E}_{p_{\theta(t)}} \left[ \exp\{f_\phi(x)\} \cdot \nabla_\phi f_\phi(x) \right] / Z_\phi. \tag{8}
\]

Note that \( Z_\phi \) is the normalization factor defined in Eq. (3). That is, fake samples from the generator are weighted by the exponentiated discriminator score when used to update the discriminator. Intuitively,
We conduct extensive experiments on three unsupervised generation tasks, including image generation, text generation, and text style transfer. The three tasks apply GANs to model different data modalities, namely image, text, and neural hidden representations. Our work instead results in a re-weighting scheme in the new context of discriminator training.

Algorithm 1 summarizes the proposed training procedure for the generator and discriminator.

Algorithm 1: GAN Training with Probability Ratio Clipping and Sampling Re-weighting

1: Initialize the generator \( p_\theta \), the discriminator \( f_\phi \), and the auxiliary classifier \( C \)
2: for \( t \leftarrow 1 \) to \( T \) do
3:   for certain number of steps do
4:     Update the discriminator \( f_\phi \) with sample re-weighting through Eqs. (7)-(8), and maintain \( f_\phi \) to have upper-bounded Lipschitz constant through, e.g., gradient penalty [18].
5:   end for
6:   for certain number of steps do
7:     Finetune the real/fake classifier \( C \) (for 1 step)
8:     Estimate probability ratio \( r_t(\theta) \) using \( C \) through Eq. (6)
9:   end for
10: end for
11: end for

3.4 Theoretical Analysis

In this section, we show that our framework produces informative gradient [57] that points the generator distribution \( p_\theta \) toward the real data distribution \( p_r \) assuming an optimal discriminator. The analysis is based on the reverse KL updates for the generator (Eq.4), while the probability ratio clipping serves as a practical emulation for the updates. We begin by adapting Proposition 1 in Gulrajani et al. [18] to our problem:

**Proposition 3.1.** Let \( p_r \) and \( p_\theta \) be two distributions in \( X \), a compact metric space. Then, there is a 1-Lipschitz function \( f^* \) which is the optimal solution of

\[
\max_{\|f_\phi\|_2 \leq 1} \mathbb{E}_{x \sim p_r} [f_\phi(x)] - \mathbb{E}_{x \sim q} [f_\phi(x)]
\]

Let \( \pi \) be the optimal coupling between \( p_r \) and \( q \), defined as the minimizer of: \( W(p_r, q) = \inf_{\pi \in \Pi(p_r, q)} \mathbb{E}_{(x, y) \sim \pi} [\|x - y\|] \) where \( \Pi(p_r, q) \) is the set of joint distributions \( \pi(x, y) \) whose marginals are \( p_r \) and \( q \), respectively. Then, if \( f^* \) is differentiable, \( \pi(x = y) = 0 \), and \( x_\tau = \tau x + (1 - \tau)y \) with \( 0 \leq \tau \leq 1 \), it holds that \( p(x, y) \sim \pi \left[ \nabla f^*(x_\tau) = \frac{y - x_\tau}{\|y - x_\tau\|} \right] = 1 \).

The proposition shows that the optimal \( f^* \) provides informative gradient [58] from \( q \) towards \( p_r \). We then generalize the conclusion to \( p_\theta \) by considering correlation between \( q \) and \( p_\theta \).

By the definition of \( q \) with respect to \( p_\theta \) in Equation (3), the support of \( p_\theta \) and \( q \) are the same; namely, given \( x \sim p_\theta, y \sim p_r \), we also have \( q(x) \neq 0 \). Therefore, for all \( x \sim p_\theta \), \( x \) is also a valid sample from \( q \), the \( f^* \) in Proposition 3.1 provides informative gradient with respect to \( x_\tau = \tau x + (1 - \tau)y, \forall \tau \in [0, 1] \):

\[
\mathbb{E}_{(x, y) \sim \pi} \left[ \nabla f^*(x_\tau) = \frac{y - x_\tau}{\|y - x_\tau\|} \right] = 1
\]

Therefore, assuming \( f^*_\phi \) is the optimal discriminator to (7), optimizing Eq. (4) can provide informative gradient that points the generator toward \( p_r \).

4 Experiments

We conduct extensive experiments on three unsupervised generation tasks, including image generation, text generation, and text style transfer. The three tasks apply GANs to model different data modalities, namely image, text, and neural hidden representations. Our approach consistently offers improvement over the state-of-the-arts on all tasks. See appendix for experimental details.
<table>
<thead>
<tr>
<th>Method</th>
<th>IS (↑)</th>
<th>FID (↓)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real data</td>
<td>11.24±.12</td>
<td>7.8</td>
</tr>
<tr>
<td>WGAN-GP (2017)</td>
<td>7.86±.08</td>
<td>-</td>
</tr>
<tr>
<td>CT-GAN (2018)</td>
<td>8.12±.12</td>
<td>-</td>
</tr>
<tr>
<td>SN-GANs (2018)</td>
<td>8.22±.05</td>
<td>21.7±.21</td>
</tr>
<tr>
<td>WGAN-ALP (2020)</td>
<td>8.34±.06</td>
<td>12.96±.35</td>
</tr>
<tr>
<td>SRNGAN (2020)</td>
<td>8.53±.04</td>
<td>19.83</td>
</tr>
<tr>
<td>AutoGAN (2019)</td>
<td>8.55±.10</td>
<td>12.42</td>
</tr>
<tr>
<td>Ours (re-weighting only)</td>
<td>8.45±.14</td>
<td>13.21±.60</td>
</tr>
<tr>
<td>Ours (full)</td>
<td>8.69±.13</td>
<td>10.70±.10</td>
</tr>
</tbody>
</table>

Table 1: CIFAR-10 results. Our method is run 3 times for average and standard deviation.

4.1 Image generation

We first use the popular CIFAR-10 benchmark for evaluation and in-depth analysis of our approach.

Setup. CIFAR-10 [29] contains 50K images of sizes 32 × 32. Following the setup in CT-GAN [51] and its public implementation 2, we use a residual architecture to implement both generator and discriminator, and also impose a Lipschitz constraint on the discriminator. For each iteration, we update both generator and discriminator for 5 times. We use Inception Score (IS) [42] and Frechet Inception Distance (FID) [20] as our evaluation metrics. Among them, IS evaluates the quality and diversity of generated images, and FID captures model issues, e.g., mode collapse [52].

Results. Table 1 reports the results on CIFAR-10. For the four latest methods, SN-GANs [34] introduced spectral normalization to stabilize the discriminator training; WGAN-ALP [43] developed an explicit Lipschitz penalty; WGAN-ALP [43] introduced a weight-normalization scheme for generalization; and AutoGAN [15] incorporated neural architecture search for the generator architecture. From Table 1, one can observe that our full approach (CT-GAN + discriminator sample re-weighting + generator probability ratio clipping) achieves the best performance, with both IS and FID significantly surpassing the baselines. These results accord with the visual results in Figure 2 where our generated samples show higher visual quality than those of the baselines. Moreover, comparison between CT-GAN and our approach with only discriminator re-weighting shows significant improvement. By further adding the probability ratio clipping to arrive our full approach, the performance in terms of both IS and FID is further improved with a large margin. The results demonstrate the effectiveness of the two components in our approach.

Figure 1 in Sec. 1 has shown the effects of the proposed approach in stabilizing the generator and discriminator training. Here we further analyze these two components. Figure 3 (left) shows the convergence curves of different GAN methods. For a fair comparison, all models use the same DCGAN architecture [40], and both our approach and WGAN-GP [18] enforce the same discriminator Lipschitz constraint. From Figure 3 (left), one can observe that our full approach surpasses our approach with only sample re-weighting, and they both converge faster and achieve a higher IS score than clipping-only. WGAN-GP, and DCGAN. Note that clipping-only does not offer a performance improvement but does not slow down training. Figure 3 (right) further looks into how the re-weighting

---

2https://github.com/igul222/improved_wgan_training
Table 2: Oracle negative log-likelihood scores (↓) on synthetic data.

<table>
<thead>
<tr>
<th>Method</th>
<th>BLEU-2 (↑)</th>
<th>BLEU-3 (↑)</th>
<th>BLEU-4 (↑)</th>
<th>BLEU-5 (↑)</th>
<th>NLL_{gen} (↓)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLE</td>
<td>0.768</td>
<td>0.473</td>
<td>0.240</td>
<td>0.126</td>
<td>2.382</td>
</tr>
<tr>
<td>SeqGAN [54]</td>
<td>0.777</td>
<td>0.491</td>
<td>0.261</td>
<td>0.138</td>
<td>2.773</td>
</tr>
<tr>
<td>LeakGAN [19]</td>
<td>0.826</td>
<td>0.645</td>
<td>0.437</td>
<td>0.272</td>
<td>2.356</td>
</tr>
<tr>
<td>RelGAN (100) [36]</td>
<td>0.881</td>
<td>0.705</td>
<td>0.501</td>
<td>0.319</td>
<td>2.482</td>
</tr>
<tr>
<td>RelGAN (1000) [36]</td>
<td>0.837</td>
<td>0.654</td>
<td>0.435</td>
<td>0.265</td>
<td>2.285</td>
</tr>
<tr>
<td>WGAN-GP [18]</td>
<td>0.872</td>
<td>0.636</td>
<td>0.379</td>
<td>0.220</td>
<td>2.209</td>
</tr>
<tr>
<td>Ours</td>
<td><strong>0.905</strong></td>
<td>0.692</td>
<td>0.470</td>
<td><strong>0.322</strong></td>
<td>2.265</td>
</tr>
</tbody>
</table>

Table 3: Results on EMNLP2017 WMT News. BLEU measures text quality and NLL_{gen} evaluates sample diversity. We copied the results of previous text GAN models from [36], where RelGAN (100) and RelGAN (1000) use different hyper-parameter for gumbel-softmax.

on fake samples can affect the discriminator training. It shows that by injecting sample re-weighting into WGAN-GP, its gradients on fake samples become more stable and show much lower variance, which also partially explains the higher training stability of discriminator in Figure 1.

4.2 Text Generation

In this section, we evaluate our approach on text generation, a task that is known to be notoriously difficult for GANs due to the discrete and sequential nature of text.

**Setup.** We implement our approach based on the RelGAN [36] architecture, a state-of-the-art GAN model for text generation. Specifically, we replace the generator and discriminator objectives in RelGAN with ours. We follow WGAN-GP [18] and impose discriminator Lipschitz constraint with gradient penalty. Same as [36], we use Gumbel-softmax approximation [26, 32] on the discrete text to enable gradient backpropagation, and the generator is initialized with maximum likelihood (MLE) pre-training. Our implementation is based on the public PyTorch code of RelGAN [36]. Same as previous studies [19, 36, 54], we evaluate our approach on both synthetic and real text datasets.

**Results on Synthetic Data.** The synthetic data consists of 10K discrete sequences generated by an oracle-LSTM with fixed parameters [54]. This setup facilitates evaluation, as the quality of generated samples can be directly measured by the negative log-likelihood (NLL) of the oracle on the samples. We use synthetic data with sequence lengths 20 and 40, respectively. Table 2 reports the results. MLE is the baseline with maximum likelihood training, whose output model is used to initialize the generators of GANs. Besides the previous text generation GANs [19, 36, 54], we also compare with WGAN-GP which uses the same neural architecture as RelGAN and ours. From Table 2, one can observe that our approach significantly outperforms all other approaches on both synthetic sets. Our improvement over RelGAN and WGAN-GP demonstrates that our proposed generator and discriminator objectives are more effective than the previous ones.

**Results on Real Data.** We then evaluate our method on the EMNLP2017 WMT News, a large real text data used for text GAN studies [19, 36]. The dataset consists of 270K/10K training/test sentences with a maximum length of 51 and a vocabulary size of 5,255. To measure the generation quality, we use the popular BLEU-n metric which measures n-gram overlap between generated and real text (n ∈ {2, 3, 4, 5}). To evaluate the diversity of generation, we use the negative log-likelihood of the generator on the real test set (NLL_{gen}) [19, 36]. From the results in Table 3, one can see that our approach shows comparable performance with the previous best model RelGAN (100) in terms of text quality (BLEU), but has better sample diversity. We use the same gumbel-softmax parameter as RelGAN (1000). Our model also achieves much higher BLEU scores than WGAN-GP (e.g., 0.322 vs. 0.220 on BLEU-5), demonstrating its ability of generating higher-quality samples.

3https://github.com/williamSYSU/TextGAN-PyTorch
Table 4: BLEU scores between model generations and human-written text on the Yelp data. We run our method for 5 times and report the average and standard deviation.

<table>
<thead>
<tr>
<th>Method</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhang et al.</td>
<td>24.48</td>
</tr>
<tr>
<td>Tian et al.</td>
<td>24.90</td>
</tr>
<tr>
<td>Subramanian et al.</td>
<td>31.20</td>
</tr>
<tr>
<td>Tikhonov et al.</td>
<td>32.82</td>
</tr>
<tr>
<td>Ours</td>
<td>33.45±.95</td>
</tr>
</tbody>
</table>

4.3 Text Style Transfer

We further apply our approach on the text style transfer task which is gaining increasing attention in NLP [23, 46]. The task aims at rewriting a sentence to modify its style (e.g., sentiment) while preserving the content. Previous work applies GANs on neural hidden states to learn disentangled representations [46, 50]. The task thus can serve as a good benchmark task for GANs, as hidden state modeling provides a new modality that differs from image and text modeling as studied above.

Setup. We follow the same experimental setting and use the same model architecture in the latest work [50]. In particular, Tikhonov et al. [50] extended the variational autoencoder based model [23, 28] by adding a latent code discriminator which eliminates stylistic information in the latent code. We replace their adversarial objectives with our proposed ones, and impose discriminator Lipschitz constraint with gradient penalty [18]. Our implementation is based on the public code [50] released in [50]. We test our approach on sentiment transfer, in which the sentiment (positive or negative) is treated as the style of the text. We use the standard Yelp review dataset [406] and the human written output text provided by [31] as the ground truth for evaluation.

Results. Following the previous work [50], we first report the BLEU score that measures the similarity of the generated samples against the human written text. Table 4 shows that our approach achieves best performance, improving the state-of-the-art result [50] from BLEU 32.82 to 33.45.

The second widely used evaluation method is to measure (1) the style accuracy by applying a pre-trained style classifier on generated text, and (2) the content preservation by computing the BLEU score between the generated text and the original input text (BLEU-X). There is often a trade-off between the two metrics. Figure 4 displays the trade-off by different models. Our results locate on the top-right corner, indicating that our approach achieves the best overall style-content trade-off.

5 Conclusion

We have presented a new training framework of GANs derived from a new variational perspective and draws on rich connections with RL-as-inference. This results in probably ratio clipping for generator updates to discourage overly large changes, and fake sample re-weighting for discriminator updates to stabilize training. Experiments show our approach achieves better results than previous best methods on image generation, text generation, and text style transfer. Our approach also shows more stable training. We are interested in exploring more connections between GANs and other learning paradigms to inspire more techniques for improved GAN training.

https://github.com/VASHibaev/text_style_transfer
www.yelp.com/dataset
Broader Impacts

This work offers a unique viewpoint on two promising fields with lots of applications and impacts: Generative Adversarial Networks and Reinforcement Learning. The improvement to image generation results may be adapted to speed up photo editing, improve scene rendering, and create more realistic simulation for robot training. Furthermore, the contribution to text generation and text style transfer can be adopted to improve the quality of machine translation, and automated news-summaries.

Nevertheless, GANs can also be applied to faking images of people and jeopardize personal identities (i.e. Deepfake). We hope that future works can counter this issue through deep-fake detection.

References


6 Appendix

6.1 Proof on the equivalence between Reverse KL Divergence and KL Divergence

We prove that optimizing $\text{KL}(p_\theta \| q)$ are equivalent to optimizing $\text{KL}(q \| p_\theta)$. This provides guarantee for the approximation that leads to (4).

**Claim:** Under the assumption that $f_\phi$ Lipschitz, $f_\phi$ is bounded because the input $x$ is bounded. Let $K$ be the Lipschitz constant of $f_\phi$, and let $c = f_\phi(0)$

$$|f_\phi(x) - c| \leq K|x - 0| = K|x|$$

We then show that $\text{KL}(p_\theta \| q)$ differ $\text{KL}(q \| p_\theta)$ by at most a constant. Since the function $f_\phi(x)$ is lower and upper-bounded. There exists $a, b$, such that $-a \leq f_\phi(x) \leq b$ for any $x$ bounded.

$$\text{KL}(q \| p_\theta) - \text{KL}(p_\theta \| q)$$

$$= \int_x \left[ q(x) \log \left( \frac{q(x)}{p_\theta(x)} \right) - p_\theta(x) \log \left( \frac{p_\theta(x)}{q(x)} \right) \right] dx$$

$$= \int_x [q(x) + p_\theta(x)] \log \left( \frac{q(x)}{p_\theta(x)} \right) dx$$

$$\geq \int_x p_\theta(x) \left[ 1 + \frac{\exp(\alpha f_\phi(x))}{Z} \right] \log \left( \frac{\exp(\alpha f_\phi(x))}{Z} \right) dx$$

$$\leq \alpha(a + b) \int_x p_\theta(x) \left[ 1 + \frac{\exp(\alpha f_\phi(x))}{Z} \right] dx$$

$$\geq 2\alpha(a + b),$$

where $\triangleq$ plugs $q^*(x) = \frac{q(x) \exp(\alpha f_\phi(x))}{Z}$; $\triangleq$ uses the fact $\log \left( \frac{\exp(\alpha f_\phi(x))}{Z} \right) = \int_x \log \left( \frac{\exp(\alpha f_\phi(x))}{Z} \right) dx = \alpha(a + b)$; $\triangleq$ uses $\int_x p_\theta(x) \frac{\exp(\alpha f_\phi(x))}{Z} dx = 1$. The above claim completes the theoretical guarantee on the reverse-KL approximation in (4).

6.2 Proof on the necessity of Lipschitz constraint on the discriminator

Although [25] shows preliminary connections between PR and GAN, the proposed PR framework does not provide informative gradient to the generator when treated as a GAN loss. Following [58], we consider the training problem when the discriminator (i.e. $f_\phi(x)$ here) is optimal: when discriminator $f^*_\phi(x)$ is optimal, then the gradient of generator $g(f_\phi(x))$ is $\nabla_{f_\phi(x)} g(f_\phi(x)) \cdot \nabla_x f^*_\phi(x)$ which could be very small due to vanished $\nabla_x f^*_\phi(x)$. In this way, it is hard to push the generated data distribution $p_\theta$ towards the targeted real distribution $p_r$. This problem also exists in ?? because

$$f^*_\phi(x) = \arg\min_{f_\phi(x)} \alpha \left[ p_r(x) f_\phi(x) - q(x) f_\phi(x) \right].$$

So if $p_r$ and $q$ are disjoint, we have

$$f^*_\phi(x) = \arg\min_{f_\phi(x)} \alpha \left[ p_r(x) f_\phi(x) - q(x) f_\phi(x) \right]$$

$$= \begin{cases} \arg\min_{f_\phi(x)} \alpha \left[ p_r(x) f_\phi(x) \right], & \text{if } x \sim p_r \\ \arg\min_{f_\phi(x)} \alpha \left[ q(x) f_\phi(x) \right], & \text{if } x \sim q. \end{cases}$$

Note that for any $x \sim p_r, f^*_\phi(x)$ is not related to $q$ and thus its gradient $\nabla f^*_\phi(x)$ also does not relate to $q$. Similarly, for any $x \sim q, \nabla f^*_\phi(x)$ does not provide any information of $p_r$. Therefore, the proposed loss in [25] cannot guarantee informative gradient [58] that pushes $q$ or $p_\theta$ towards to $p_r$.

6.3 Experiments: More Details and Results

6.3.1 Binary classifier for probability ratio clipping

For the image generation and text generation, the binary classifier $C$ in Eq.6 has the same architecture as the discriminator except an additional Sigmoid activation at the output layer. The binary
classifier is trained with real and fake mini-batches alongside the generator, and requires no additional loops. We select the clipping parameter $\epsilon$ from $\{0.2, 0.4\}$, as they are typically used in PPO.

In addition in the task of image generation, we observe similar overall performance between training on raw inputs from the generator/dataset and training on input features from the first residual block of the discriminator ($D$), thus further reducing the computational overhead of the binary classifier.

### 6.3.2 Image Generation on CIFAR-10

We translate the code provided by Wei et al. [51] into Pytorch to conduct our experiments. We use the same architecture: a residual architecture for both generator and discriminator, and enforcing Lipschitz constraint on the discriminator in the same way as CT-GAN [51]. During training, we interleave 5 generator iterations with 5 discriminator iterations.

We optimize the generator and discriminators with Adam (Generator lr: $5e^{-5}$, Discriminator lr: $1e^{-4}$, betas: $(0.0, 0.9)$). We set the clipping threshold $\epsilon := 0.4$ for the surrogate loss and we linearly anneal the learning rate with respect to the number of training epochs.

#### Discriminator sample re-weighting stabilizes DCGAN

We quantitatively evaluate the effect of discriminator re-weighted sampling by comparing DCGAN [40] against DCGAN with discriminator re-weighting. Starting from the DCGAN architecture and hyper-parameters, we run 200 random configurations of learning rate, batch size, non-linearity (ReLU/LeakyReLU), and base filter count (32, 64). Results are summarized in Table 5. DCGANs trained with re-weighted sampling has significantly less collapse rate, and achieves better overall performance in terms of Inception Score.

These results well demonstrate the effectiveness of the proposed discriminator re-weighted sampling mechanism.

<table>
<thead>
<tr>
<th>Method</th>
<th>Collapse rate</th>
<th>Avg IS</th>
<th>Best IS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCGAN</td>
<td>52.4%</td>
<td>4.2</td>
<td>6.1</td>
</tr>
<tr>
<td>DCGAN + Re-weighting</td>
<td>30.2%</td>
<td>5.1</td>
<td>6.7</td>
</tr>
</tbody>
</table>

Table 5: Outcomes of 200 trials with random configurations. The performance of the models are measured through Inception score. We identify training collapse when the average discriminator loss over 2000 batches is below $1e^{-20}$ or above $1 - 1e^{-20}$. DCGAN re-weighted with our loss has lower collapse rate and higher average performance.

#### Discriminator re-weighted samples

To provide an illustration of how discriminator weights can help the discriminator concentrate on the fake samples of better quality during the training phase, in Figure 5 we plot the fake samples of a trained ResNet model alongside their corresponding discriminator weights.

![Figure 5: One batch of generated images together with their corresponding softmax discriminator weights. The more photo-realistic images (columns 2, 3, 5, 8) receive higher discriminator weights. In this batch, the generator will be influenced more by gradients from the better-quality samples above.

#### Clipped surrogate objective

One unique benefit of the clipped surrogate objective is that it allows our model to obtain an estimate of the effectiveness of the discriminator, which then enables us to follow a curriculum that takes more than one ($n_g$) generator steps per ($n_c$) critic steps. In practice, setting $n_g = n_c = 5$ achieves good quality, which also allows us to take 5 times more generator steps than prior works [2, 18, 34, 51] with the same number of discriminator iterations. Table 1 shows the improvement enabled by applying the surrogate objective.

#### Generated samples

![Figure 6 shows more image samples by our model.](github.com/biuyq/CT-GAN)
6.3.3 Text Generation

We build upon the Pytorch implementation of RelGAN. We use the exact same model architecture as provided in the code, and enforce Lipschitz constraint on the discriminator in the same way as in WGAN-GP. During training, we interleave 5 generator iterations with 5 discriminator iterations. We use Adam optimizer (generator lr: 1e-4, discriminator lr: 3e-4). We set the clipping threshold $\epsilon = 0.2$ for the surrogate loss and we linearly anneal the learning rate with respect to the number of training epochs.

6.3.4 Text Style Transfer

We build upon the Texar-TensorFlow style-transfer model by Tikhonov et al. We use the exact same model architecture and hyper-parameters as provided in the code, and enforce Lipschitz constraint on the discriminator in the same way as WGAN-GP. In addition, we replace the discriminator $D$ in Figure by our loss with an auxiliary linear style classifier as in Odena et al.
Figure 7: Model architecture from [50], where the style discriminator ($D$) is a structured constraint the generator optimize against. A latent code discriminator ensure the independence between semantic part of the latent representation and the style of the text. Blue dashed arrows denote additional independence constraints of latent representation and controlled attribute, see [50] for the details.

We did not apply the surrogate loss to approximate the KL divergence, but relied on gradient clipping on the generator.