

Loss-Sensitive Generative Adversarial Networks on Lipschitz Densities

Guo-Jun Qi

Abstract—In this paper, we present a novel Loss-Sensitive GAN (LS-GAN) that learns a loss function to separate generated samples from their real examples. An important property of the LS-GAN is it allows the generator to focus on improving poor data points that are far apart from real examples rather than wasting efforts on those samples that have already been well generated, and thus can improve the overall quality of generated samples. The theoretical analysis also shows that the LS-GAN can generate samples following the true data density. In particular, we present a regularity condition on the underlying data density, which allows us to use a class of Lipschitz losses and generators to model the LS-GAN. It relaxes the assumption that the classic GAN should have infinite modeling capacity to obtain the similar theoretical guarantee. Furthermore, we derive a non-parametric solution that characterizes the upper and lower bounds of the losses learned by the LS-GAN, both of which are piecewise linear and have non-vanishing gradient almost everywhere. Therefore, there should be sufficient gradient to update the generator of the LS-GAN even if the loss function is optimized, relieving the vanishing gradient problem in the classic GAN and making it easier to train the LS-GAN generator. We also generalize the unsupervised LS-GAN to a conditional model generating samples based on given conditions, and show its applications in both supervised and semi-supervised learning problems. The experiment results demonstrate competitive performances on both classification and generation tasks.

Index Terms—Loss-Sensitive GAN, Lipschitz density, image generation and classification

1 INTRODUCTION

A Classic Generative Adversarial Net (GAN) [1] learns a discriminator and a generator simultaneously by playing a two-player minimax game to generate samples following the underlying data density. For this purpose, the discriminator is trained to distinguish real samples from those generated by the generator, which in turn guides the generator to produce samples that can make the discriminator believe they are real.

The family of GAN models have demonstrated very impressive performances on synthesizing a wide range of structured data, as diverse as images [2], videos [3], music [4] and even poems [5]. Take image synthesis as an example. On one hand, the discriminator seeks to learn the probability of a sample being a photo-realistic image. It treats natural image examples as positive examples, while the images produced by a paired generator as negative examples. Meanwhile, the generator aims to produce images that can make the discriminator believe they are real. A minimax optimization problem is solved to jointly optimize the discriminator and the generator.

Here, a dyadic treatment of real and generated data as positive and negative examples may oversimplify the problem of learning a GAN model. Actually, as the generator improves, its generated samples would become more and more closer to the manifold of real examples; however, they are still being treated as negative examples to train the discriminator in the classic GAN. This could lead to an over-pessimistic discriminator characterizing the boundary between real and unreal samples. It would in turn limit the ability of learning a better generator that relies on an

exact discriminator to capture the difference between real and unreal examples.

In addition, from a theoretical perspective, the analysis behind the GAN makes a non-parametric assumption that the model has infinite modeling capacity [1] in order to prove the density of generated samples matches the underlying data density we wish to estimate. This is a too strong assumption to hold: even a very deep network could not assume infinite modeling capacity to map any given input to an arbitrarily desired output. Even worse, a generative model with unlimited capacity would also be the cause of vanishing gradient in the classic GAN. As proved in [6], a discriminator with infinite ability of separating real from generated samples will lead to a constant Jensen-Shannon (JS) divergence between the generated density and the true data density that have no or negligible overlap, and thus minimizing this cost JS divergence suffers from vanishing gradient that makes it impossible to update the generator as the discriminator is updated to optimality.

Moreover, the GAN with infinite modeling ability could also suffer from severe overfitting problem, and this is probably the reason for a collapsed generator that is stuck in producing the same data point since such a model would be powerful enough to aggressively push its generated samples close to the densest mode of the underlying density. The phenomenon has been observed in literature [2], [7], and a properly regularized learning objective is preferred to avoid the mode collapse.

In this paper, we attempt to develop theory and models without such an assumption, which yields a novel Loss-Sensitive GAN (LS-GAN). Specifically, we introduce a loss function to quantify the quality of generated samples. Then a constraint is imposed to train the LS-GAN so that the loss of a real sample should be smaller than that of a generated

• G.-J. Qi was with the Department of Computer Science, University of Central Florida, Orlando, FL, 32816.
E-mail: guojun.qi@ucf.edu

counterpart by an unfixed margin that depends on how close they are to each other in a metric space. In this way, if a generated sample is already very close to a real example, the margin between their losses could vanish. This allows the model to focus on improving poor samples rather than wasting efforts on those samples that have already been well generated with satisfactory quality, thereby improving overall quality of generation results.

We also develop a new theory to analyze the proposed LS-GAN on Lipschitz densities. We note that the reason of having to assume “infinite modeling capacity” in the classic GAN is due to its ambitious goal to model an arbitrary data density without imposing any biases. However, a general principle in learning theory, no free lunch theorem [8], prefers “biased learning approaches” with suitable priors specifying the problems to be solved. This prompts us to focus on a specific class of Lipschitz densities to model the underlying data. It contains a large family of real-world distributions, where the data density does not change abruptly over points that are close to one another. By equipping the Lipschitz densities with the distance metric specifying the loss margin in the LS-GAN, we prove the resulting data density from the LS-GAN exactly matches the underlying data density that Lipschitz continuous.

We further present a non-parametric solution to the LS-GAN. It does not rely on any parametric forms of functions, thereby searching the whole space of Lipschitz functions for the optimal loss function. This non-parametric solution characterizes both the upper and lower bounds of an optimal loss function. Both bounds are piecewise linear and have non-vanishing gradient. This ensures to update the LS-GAN generator even if the loss function has been fully optimized, avoiding the vanishing gradient problem to train the generator in the classic GAN.

Moreover, we generalize the model to a Conditional LS-GAN (CLS-GAN) that can generate samples based on given conditions. Specifically, considering different classes as generative conditions, the learned loss function can be used as a classifier for both supervised and semi-supervised learning. The advantage of such a classifier lies in its intrinsic ability of exploring generated examples to reveal unseen variations for different classes. Experiment results demonstrate competitive performance of the CLS-GAN classifier as compared with the state-of-the-art models.

The remainder of this paper is organized as follows. Section 2 reviews the related work and summarizes our contributions. We will present the proposed LS-GAN in Section 3. In Section 4, we will analyze the LS-GAN, showing that its generated samples follow the underlying data density even with a class of Lipschitz losses and generators. We will discuss the algorithm details in Section 5, and show that the model can be generalized to take an input condition to generate data in Section 6. Experiment results are presented in Section 7, followed by the conclusion in Section 8.

2 RELATED WORK AND OUR CONTRIBUTIONS

It has been a long-term goal to enable synthesis of highly structured data such as images and videos.

Deep generative models, especially the Generative Adversarial Net (GAN) [1], have attracted many attentions recently due to their demonstrated abilities of generating real samples following the underlying data densities. In particular, the GAN attempts to learn a pair of discriminator and generator by playing a maximin game to seek an equilibrium, in which the discriminator is trained by distinguishing real samples from generated ones and the generator is optimized to produce samples that can fool the discriminator.

A family of GAN architectures have been proposed to implement this idea. For example, recent progresses [2], [7] have shown impressive performances on synthesizing photo-realistic images by constructing multiple strided and fractional-strided convolutional layers for discriminators and generators. On the contrary, [9] proposed to use a Laplacian pyramid to produce high-quality images by iteratively adding multiple layers of noises at different resolutions. [10] presented to train a recurrent generative model by using adversarial training to unroll gradient-based optimizations to create high quality images.

In addition to designing different GAN networks, research efforts have been made to train the GAN by different criteria. For example, [11] presented an energy-based GAN by minimizing an energy function to learn an optimal discriminator, and an auto-encoder structured discriminator is presented to compute the energy. The authors also present a theoretical analysis by showing this variant of GAN can generate samples whose density can recover the underlying true data density. However, it still needs to assume the model has infinite modeling capacity to prove the result in a non-parametric fashion. This is probably due to the use of a fixed margin to separate generated samples from training examples. This is in contrast to the use of a distance metric in the proposed LS-GAN to specify data-dependent margins under the Lipschitz density assumption. In addition, [12] presented to analyze the GAN from information theoretical perspective, and they seek to minimize the variational estimate of f-divergence, and show that the classic GAN is included as a special case of f-GAN. In contrast, InfoGAN [13] proposed another information-theoretic GAN to learn disentangled representations capturing various latent concepts and factors in generating samples.

Besides the class of GANs, there exist other models that also attempt to generate natural images. For example, [14] rendered images by matching features in a convolutional network with respect to reference images. [15] used deconvolutional network to render 3D chair models in various styles and viewpoints. [16] introduced a deep recurrent neural network architecture for image generation with a sequence of variational auto-encoders to iteratively construct complex images.

Recent efforts have also been made on leveraging the learned representations by deep generative networks to improve the classification accuracy when it is too difficult or expensive to label sufficient training examples. For example, [17] presented variational auto-encoders [18] by combining deep generative models and approximate variational inference to explore both labeled and unlabeled data. [2] treated the samples from the GAN generator as a new class, and explore unlabeled examples by assigning them to a class

different from the new one. [19] proposed to train a ladder network [20] by minimizing the sum of supervised and unsupervised cost functions through back-propagation, which avoids the conventional layer-wise pre-training approach. [21] presented an approach to learning a discriminative classifier by trading-off mutual information between observed examples and their predicted classes against an adversarial generative model. These methods have shown promising results for classification tasks by leveraging deep generative networks and/or their generated samples.

In this paper, we seek to develop models and algorithms that are both theoretically sound and practically competitive for data generation and classification tasks. Our contributions are summarized below.

- We propose a Loss-Sensitive GAN (LS-GAN) model to produce high-quality samples. The LS-GAN learns a loss function to quantify the quality of generated samples. The loss of a real example should be smaller than that of a generated sample by a margin characterized by their distance in a metric space. The well generated samples close to real examples do not need to be treated as negative examples anymore so that more efforts can be focused on improving the quality of poor samples.
- We also generalize LS-GAN to a conditional version that shares the same theoretical merit as the LS-GAN but can generate samples aligned with designed conditions. Specifically, we consider to specify sample classes as conditions, and this model can produce multiple classes of examples that capture intra-class variations. This yields a classifier using the learned loss function and exploring the generated samples to improve classification accuracy.
- We develop a new theory that introduces Lipschitz assumption to characterize underlying data densities. We will prove the LS-GAN can reveal the true density even with limited modeling ability of bounded Lipschitz constant on the generators and loss functions. This is a nontrivial relaxation of the assumption on the classic GAN that assumes infinite modeling ability in the theoretical analysis. Moreover, we also characterize the optimal loss function by deriving its lower and upper bounds, and show how these piecewise linear bounds suggest the learned loss function by the LS-GAN avoids from being saturated that causes vanishing gradient problem in training the generator of the classic GAN.

3 LOSS-SENSITIVE GAN

In the proposed LS-GAN, we abandon to learn a discriminator that uses a probability to characterize the likelihood of real samples. Instead, we introduce a loss function $L_\theta(\mathbf{x})$ to distinguish real and generated samples by the assumption that a real example should have a smaller loss than a generated sample.

Formally, consider a real example \mathbf{x} and a generated one $G_\phi(\mathbf{z})$ with $\mathbf{z} \sim P_z(\mathbf{z})$. The loss function can be trained with the following constraint:

$$L_\theta(\mathbf{x}) \leq L_\theta(G_\phi(\mathbf{z})) - \Delta(\mathbf{x}, G_\phi(\mathbf{z})) \quad (1)$$

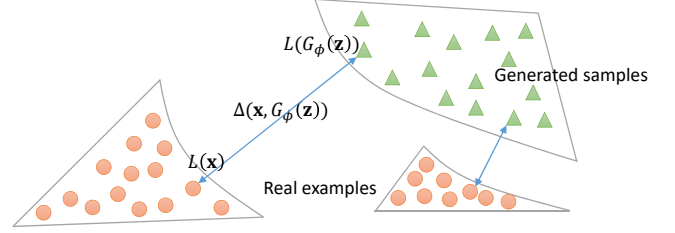


Fig. 1. Illustration of the idea behind LS-GAN. A margin is enforced to separate real samples from generated counterparts. The margin is not fixed to a constant. Instead it is data-dependent, which could vanish as the generator improves to produce better and better samples. We assume the density of real samples is Lipschitz as to prove the theoretical results.

where $\Delta(\mathbf{x}, G_\phi(\mathbf{z}))$ measures the difference between \mathbf{x} and $G_\phi(\mathbf{z})$. This constraint requires a real sample be separated from a generated counterpart in terms of their losses by at least a margin of $\Delta(\mathbf{x}, G_\phi(\mathbf{z}))$. Figure 1 illustrates this idea.

It is noteworthy that the margin is not fixed to a constant. Instead, it is data-dependent, which could vanish as the generator is gradually improved to produce better samples as they become closer to real examples. For example, one can choose the ℓ_p -distance $\|\mathbf{x} - G_\phi(\mathbf{z})\|_p$ as the margin. This allows the model to focus on improving the poor samples still far away from real examples rather than wasting efforts on those that are already well generated. In the theoretical analysis, such a data-dependent margin will also be used to specify a Lipschitz condition, which plays a critical role in guaranteeing generated samples follow the underlying data density.

Now let us relax the above hard constraint by introducing a slack variable $\xi_{\mathbf{x}, \mathbf{z}}$

$$L_\theta(\mathbf{x}) - \xi_{\mathbf{x}, \mathbf{z}} \leq L_\theta(G_\phi(\mathbf{z})) - \Delta(\mathbf{x}, G_\phi(\mathbf{z})) \quad (2)$$

$$\xi_{\mathbf{x}, \mathbf{z}} \geq 0 \quad (3)$$

where the slack variable would be nonzero when a violation of the constraint occurs.

Therefore, with a fixed generator G_ϕ , the loss function parameterized with θ can be trained by

$$\min_{\theta} \mathbb{E}_{\mathbf{x} \sim P_{data}(\mathbf{x})} L_\theta(\mathbf{x}) + \lambda \mathbb{E}_{\substack{\mathbf{x} \sim P_{data}(\mathbf{x}) \\ \mathbf{z} \sim P_z(\mathbf{z})}} \xi_{\mathbf{x}, \mathbf{z}} \quad (4)$$

$$\text{s.t., } L_\theta(\mathbf{x}) - \xi_{\mathbf{x}, \mathbf{z}} \leq L_\theta(G_\phi(\mathbf{z})) - \Delta(\mathbf{x}, G_\phi(\mathbf{z})) \\ \xi_{\mathbf{x}, \mathbf{z}} \geq 0$$

where λ is a positive balancing parameter, and $P_{data}(\mathbf{x})$ is the data distribution for real samples. The first term minimizes the expected loss function over data distribution since a smaller value of loss function is preferred on real samples. The second term is the expected error caused by the violation of the constraint. Without loss of generality, we require the loss function should be nonnegative. Later we will show that this nonnegative requirement can be dropped in some case.

On the other hand, for a fixed loss function L_θ , one can solve the following minimization problem to find an optimal generator.

$$\min_{\phi} \mathbb{E}_{\substack{\mathbf{x} \sim P_{data}(\mathbf{x}) \\ \mathbf{z} \sim P_z(\mathbf{z})}} L_\theta(G_\phi(\mathbf{z})) \quad (5)$$

In summary, L_θ and G_ϕ are alternately optimized by solving a Nash equilibrium (θ^*, ϕ^*) such that θ^* minimizes

$$S(\theta, \phi^*) = \mathbb{E}_{\mathbf{x} \sim P_{data}(\mathbf{x})} L_\theta(\mathbf{x}) + \lambda \mathbb{E}_{\substack{\mathbf{x} \sim P_{data}(\mathbf{x}) \\ \mathbf{z}_G \sim P_{G^*}(\mathbf{z}_G)}} (\Delta(\mathbf{x}, \mathbf{z}_G) + L_\theta(\mathbf{x}) - L_\theta(\mathbf{z}_G))_+ \quad (6)$$

which is an equivalent compact form of (4) with $(a)_+ = \max(a, 0)$, and ϕ^* minimizes

$$T(\theta^*, \phi) = \mathbb{E}_{\mathbf{z}_G \sim P_G(\mathbf{z}_G)} L_{\theta^*}(\mathbf{z}_G) \quad (7)$$

where $P_G(\mathbf{z}_G)$ is the density of samples generated by $G_\phi(\mathbf{z})$.

4 THEORETICAL ANALYSIS

Suppose (θ^*, ϕ^*) is a Nash equilibrium that jointly solves (6) and (7). We will show that as $\lambda \rightarrow +\infty$, the density distribution P_{G^*} of the samples generated by G_{ϕ^*} will converge to the underlying data density P_{data} .

To prove this result, we need the following definition.

Definition. For any two samples \mathbf{x} and \mathbf{z} , the loss function $F(\mathbf{x})$ is Lipschitz continuous with respect to a distance metric Δ if

$$|F(\mathbf{x}) - F(\mathbf{z})| \leq \kappa \Delta(\mathbf{x}, \mathbf{z})$$

with a bounded Lipschitz constant κ , i.e., $\kappa < +\infty$.

To prove our main result, we assume the following regularity condition on the underlying data density.

Assumption 1. The data density P_{data} is supported in a compact set, and it is Lipschitz continuous.

The set of Lipschitz densities on a compact support contain a large family of distributions that are dense in the space of continuous densities. For example, the density of natural images can be consider as Lipschitz continuous, as the densities of two similar images in a neighborhood are unlikely to change abruptly. Moreover, one can restrict the image density in a compact support as an image has bounded pixel values on $[0, 255]$.

This is contrary to the analysis of the classic GAN, where one must assume both discriminator and generator have infinite modeling ability to prove P_{G^*} equals P_{data} . The Lipschitz assumption on the data density allows us to relax such a strong assumption to Lipschitz loss function L_θ and generator density P_G . This results in the following lemma relating P_{G^*} to P_{data} .

Lemma 1. Under Assumption 1, given a Nash equilibrium (θ^*, ϕ^*) such that P_{G^*} is Lipschitz continuous, we have

$$\int_{\mathbf{x}} |P_{data}(\mathbf{x}) - P_{G^*}(\mathbf{x})| d\mathbf{x} \leq \frac{2}{\lambda}$$

Thus, $P_{G^*}(\mathbf{x})$ converges to $P_{data}(\mathbf{x})$ as $\lambda \rightarrow +\infty$.

The proof of this lemma is given in the appendix.

Remark 1. From this theorem, we find that by allowing λ infinitely large, the learned density $P_{G^*}(\mathbf{x})$ should exactly match the data density $P_{data}(\mathbf{x})$. In other words, we can simply disregard the first loss minimization term in (6) as it plays no role as $\lambda \rightarrow +\infty$. It is also not hard to see that if we disregard the first minimization term, the requirement that the loss function L_θ is nonnegative is not needed anymore to prove the above theorem,

and this gives us more flexibility in designing loss function for the LS-GAN.

However, the reason that we still keep the loss minimization term in the formulation will become clear after we develop the conditional LS-GAN later.

Now we can show the existence of Nash equilibrium such that both the loss function L_θ and the density $P_G(\mathbf{z}_G)$ of generated samples are Lipschitz.

Let \mathcal{F}_κ be the class of functions with a bounded Lipschitz constant κ . It is not difficult to show that the space \mathcal{F}_κ is convex and compact if these functions are supported in a compact set¹. In addition, both $S(\theta, \phi)$ and $T(\theta, \phi)$ are convex functions in L_θ and in $P_G(\mathbf{z}_G)$. These guarantee the existence of a Nash equilibrium (θ^*, ϕ^*) with both L_{θ^*} and P_{G^*} in \mathcal{F}_κ , following the proof of the classic mixed-strategy game theory by applying Kakutani fixed-point theorem [22]. Thus, we have the following lemma.

Lemma 2. Under Assumption 1, there exists a Nash equilibrium (θ^*, ϕ^*) such that L_{θ^*} and P_{G^*} are Lipschitz.

Putting the above two lemmas together, we have the following theorem.

Theorem 1. Under Assumption 1, a Nash equilibrium (θ^*, ϕ^*) exists such that

- (i) L_{θ^*} and P_{G^*} are Lipschitz.
- (ii) $\int_{\mathbf{x}} |P_{data}(\mathbf{x}) - P_{G^*}(\mathbf{x})| d\mathbf{x} \leq \frac{2}{\lambda} \rightarrow 0$, as $\lambda \rightarrow +\infty$;
- (iii) $P_{data}(\mathbf{x}) \geq \frac{\lambda}{1+\lambda} P_{G^*}(\mathbf{x})$.

5 ALGORITHM

The minimization problems (6) and (7) can be rewritten by replacing the expectation with a given set of examples $\mathcal{X}_n = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ and the noise vectors $\mathcal{Z}_m = \{\mathbf{z}_1, \dots, \mathbf{z}_m\}$ drawn from a distribution $P_z(\mathbf{z})$.

This results in the following two problems.

$$\begin{aligned} \min_{\theta} S_{n,m}(\phi^*, \theta) &\triangleq \frac{1}{n} \sum_{i=1}^n L_\theta(\mathbf{x}_i) \\ &+ \frac{\lambda}{nm} \sum_{i,j=1}^{n,m} (\Delta(\mathbf{x}_i, G_{\phi^*}(\mathbf{z}_j)) + L_\theta(\mathbf{x}_i) - L_\theta(G_{\phi^*}(\mathbf{z}_j)))_+ \end{aligned} \quad (8)$$

and

$$\min_{\phi} T_k(\theta^*, \phi) = \frac{1}{k} \sum_{i=1}^k L_{\theta^*}(G_\phi(\mathbf{z}'_i)) \quad (9)$$

The random vectors $\mathcal{Z}'_k = \{\mathbf{z}'_i | i = 1, \dots, k\}$ used in (9) can be different from \mathcal{Z}_m used in (8).

The loss function and the generator can be learned by alternating between these two problems over mini-batches. In each mini-batch, a set \mathcal{Z}_m of random noises are sampled from a prior distribution $P_z(\mathbf{z})$, along with a subset of real samples from the training set \mathcal{X}_n . Then, the loss function is updated by descending the gradient of (8), and the generator is updated by minimizing (9) with a set of random vectors \mathcal{Z}'_k sampled from $P_z(\mathbf{z})$. After the generator G_{ϕ^*} has been updated, the data points $\mathbf{x}^{(n+j)} = G_{\phi^*}(\mathbf{z}_j)$

1. For example, the space of natural images is compact as their pixel values are restricted to a compact range of $[0, 255]$.

of generated samples will also be updated. Algorithm 1 summarizes the learning algorithm for the LS-GAN.

5.1 Non-Parametric Characterization of Loss functions

Now let us characterize the optimal loss functions based on the objective (8), which will provide us an insight into the LS-GAN algorithm.

We generalize the non-parametric maximum likelihood method in [23] and consider non-parametric solutions to the optimal loss function by minimizing (8) over the whole class of Lipschitz loss functions.

Let $\mathbf{x}^{(1)} = \mathbf{x}_1, \mathbf{x}^{(2)} = \mathbf{x}_2, \dots, \mathbf{x}^{(n)} = \mathbf{x}_n, \mathbf{x}^{(n+1)} = G_{\phi^*}(\mathbf{z}_1), \dots, \mathbf{x}^{(n+m)} = G_{\phi^*}(\mathbf{z}_m)$, i.e., the first n data points are real examples and the rest m are generated samples. Then we have the following theorem.

Theorem 2. *The following functions \hat{L}_{θ^*} and \tilde{L}_{θ^*} both minimize $S_{n,m}(\theta, \phi^*)$ in \mathcal{F}_κ :*

$$\begin{aligned}\hat{L}_{\theta^*}(\mathbf{x}) &= \max_{1 \leq i \leq n+m} \{ (l_i^* - \kappa \Delta(\mathbf{x}, \mathbf{x}^{(i)}))_+ \}, \\ \tilde{L}_{\theta^*}(\mathbf{x}) &= \min_{1 \leq i \leq n+m} \{ l_i^* + \kappa \Delta(\mathbf{x}, \mathbf{x}^{(i)}) \}\end{aligned}\quad (10)$$

with the same group of parameters $\theta^* = [l_1^*, \dots, l_{n+m}^*] \in \mathbb{R}^{n+m}$. They are supported in the convex hull of $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(n+m)}\}$, and we have

$$\hat{L}_{\theta^*}(\mathbf{x}^{(i)}) = \tilde{L}_{\theta^*}(\mathbf{x}^{(i)}) = l_i^*$$

for $i = 1, \dots, n+m$, i.e., their values coincide on $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(n+m)}\}$.

The proof of this theorem is given in the appendix.

From the theorem, it is not hard to show that any convex combination of these two forms attains the same value of $S_{n,m}$, and is also its global minimizer. Thus, we have the following corollary.

Corollary 1. *All the functions in*

$$\mathcal{L}_{\theta^*} = \{ \gamma \hat{L}_{\theta^*} + (1 - \gamma) \tilde{L}_{\theta^*} | 0 \leq \gamma \leq 1 \} \subset \mathcal{F}_\kappa$$

are the global minimizer of $S_{n,m}$ in \mathcal{F}_κ .

This shows that the global minimizer is not unique. Moreover, through the proof of Theorem 2, one can find that $\tilde{L}_{\theta^*}(\mathbf{x})$ and $\hat{L}_{\theta^*}(\mathbf{x})$ are the upper and lower bound of any optimal loss function solution to the problem (8). In particular, we have the following corollary.

Corollary 2. *For any $L_{\theta^*}(\mathbf{x}) \in \mathcal{F}_\kappa$ that minimizes $S_{n,m}$, the corresponding $\hat{L}_{\theta^*}(\mathbf{x})$ and $\tilde{L}_{\theta^*}(\mathbf{x})$ are the lower and upper bounds of $L_{\theta^*}(\mathbf{x})$, i.e.,*

$$\hat{L}_{\theta^*}(\mathbf{x}) \leq L_{\theta^*}(\mathbf{x}) \leq \tilde{L}_{\theta^*}(\mathbf{x})$$

The proof is given in Appendix B.

The parameters $\theta^* = [l_1^*, \dots, l_{n+m}^*]$ in (10) can be sought by minimizing

$$\begin{aligned}S_{n,m}(\phi^*, \theta) &\triangleq \frac{1}{n} \sum_{i=1}^n l_i + \frac{\lambda}{nm} \sum_{i,j=1}^{n,m} (\Delta_{i,n+j} + l_i - l_{n+j})_+ \\ \text{s.t.}, & |l_i - l_{i'}| \leq \kappa \Delta(\mathbf{x}^{(i)}, \mathbf{x}^{(i')}) \\ & l_i \geq 0, \quad i, i' = 1, \dots, n+m\end{aligned}\quad (11)$$

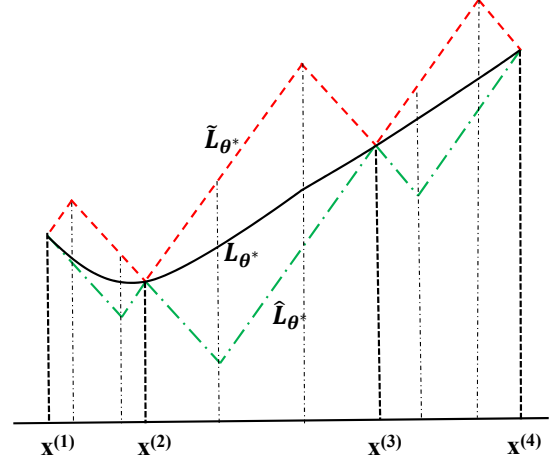


Fig. 2. Comparison between two optimal loss functions \tilde{L}_{θ^*} and \hat{L}_{θ^*} for LS-GAN. They are upper and lower bounds of the class of optimal loss functions L_{θ^*} to Problem (8). Both the upper and lower bounds are piece-wise linear, and have non-vanishing gradient almost everywhere, and non-vanishing gradient makes it amenable to update the generator of LS-GAN by descending (9) in its negative gradient.

where $\Delta_{i,j}$ is short for $\Delta(\mathbf{x}^{(i)}, \mathbf{x}^{(n+i)})$, and the constraints are imposed to ensure the learned loss functions stay in \mathcal{F}_κ . With a greater value of κ , a larger class of loss function will be sought. Thus, one can control the modeling ability of the loss function by setting a proper value to κ .

Problem (11) is a typical linear programming problem. In principle, one can solve this problem to obtain a non-parametric loss function for the LS-GAN. Unfortunately, it consists of a large number of constraints, whose scale is at an order of $\binom{n+m}{2}$. This prevents us from using (11) directly to solve an optimal LS-GAN model with a very large number of training examples.

However, a more tractable solution is to use a parameterized network to solve this non-parametric optimization problem (8) constrained in \mathcal{L}_κ , and this is exactly the gradient descent method adopted in Algorithm 1 that iteratively updates parameterized L_θ and G_ϕ . To ensure the loss function to have a bounded Lipschitz constant, one can use weight decay to avoid too large value of network weights, or directly clamp the weights to a bounded area. In this paper, we adopt weight decay and find it works well with the LS-GAN model in experiments.

Although the non-parametric solution cannot be solved directly, it is valuable in characterizing the loss function learned by a deep network, which can shed some light on how the LS-GAN is trained. It is well known that the training of the classic GAN generator suffers from vanishing gradient problem as the discriminator can be optimized very quickly. Recent study [24] has revealed that this is caused by using the Jensen-Shannon (JS) distance that becomes locally saturated and gets vanishing gradient to train the GAN generator if the discriminator is over-trained. Similar problem has also been found in the energy-based GAN (EBGAN) [11] as it minimizes the total variation that is not continuous or (sub-)differentiable if the corresponding discriminator is fully optimized [24].

On the contrary, as revealed in Theorem 2 and illustrated

in Figure 2, both the upper and lower bounds of the optimal LS-GAN loss functions are piece-wise linear (in terms of $\Delta(\mathbf{x}, \mathbf{x}^{(i)})$ that defines the Lipschitz continuity), and have non-vanishing gradient almost everywhere. This indicates that an optimal loss function that is properly sought in \mathcal{L}_κ as shown in Figure 2 is unlikely to saturate between these two piecewise linear bounds, and thus it should be able to provide sufficient gradient to update the generator by descending (9) even if it has been trained to be optimal.

5.2 Comparison with Wasserstein GAN

We notice that the recently proposed Wasserstein GAN (WGAN) [24] uses the Earth-Mover (EM) distance to address the vanishing gradient and saturated JS distance problems in the classic GAN by showing the EM distance is continuous and differentiable almost everywhere. While the LS-GAN and the WGAN address these problems by different approaches that are independently developed, both turn out to use the Lipschitz constraint to learn the loss function of the LS-GAN and the critic of the WGAN respectively. This constraint plays vital but different roles in the two models. In the LS-GAN, the Lipschitz constraint on the loss function naturally arises from the Lipschitz regularity imposed on the data density. Under this regularity condition, we have proved in Theorem 1 that the density of generated samples is Lipschitz and consistent with the underlying data density. On the contrary, the WGAN introduces the Lipschitz constraint from the Kantorovich-Rubinstein duality of the EM distance but it is unclear in [24] if the WGAN is also based on the same Lipschitz regularity on the underlying data density.

Here we assert that the WGAN also models an underlying Lipschitz density. To prove this, we restate the WGAN as follows. The WGAN seeks to find a critic f_w^* and a generator g_ϕ^* such that

$$f_w^* = \arg \max_{f_w \in \mathcal{F}_1} U(f_w, g_\phi^*) \quad (12)$$

$$\triangleq \mathbb{E}_{\mathbf{x} \sim P_{data}} [f_w(\mathbf{x})] - \mathbb{E}_{\mathbf{z} \sim P_{\mathbf{z}}} [f_w(g_\phi^*(\mathbf{z}))]$$

and

$$g_\phi^* = \arg \max V(f_w^*, g_\phi) \triangleq \mathbb{E}_{\mathbf{z} \sim P_{\mathbf{z}}} [f_w^*(g_\phi(\mathbf{z}))] \quad (13)$$

Let $P_{g_\phi^*}$ be the density of samples generated by g_ϕ^* . Then, we prove the following lemma about the WGAN in Appendix C.

Lemma 3. *Under Assumption 1, given an optimal solution (f_w^*, g_ϕ^*) to the WGAN such that $P_{g_\phi^*}$ is Lipschitz, we have*

$$\int_{\mathbf{x}} |P_{data}(\mathbf{x}) - P_{g_\phi^*}(\mathbf{x})| d\mathbf{x} = 0$$

This lemma shows both the LS-GAN and the WGAN are based on the same Lipschitz regularity condition.

Although both methods are derived from very different perspectives, it is interesting to make a comparison between their respective forms. Formally, the WGAN seeks to maximize the difference between the first-order moments of f_w under the densities of real and generated examples. In this sense, the WGAN can be considered as a kind of *first-order moment* method. Numerically, as shown in the second term of Eq. (12), f_w tends to be minimized to be arbitrarily small

Algorithm 1 Learning algorithm for LS-GAN.

Input: n data examples \mathcal{X}_n , and λ .
for a number of iterations **do**
 for a number of steps **do**
 $\backslash \backslash$ Update the loss function over minibatches;
 Sample a minibatch from \mathcal{X}_n ;
 Sample a minibatch from \mathcal{Z}_m ;
 Update the loss function L_θ by descending the gradient of (8) with weight decay over the minibatches;
 end for
 Sample a set of \mathcal{Z}_k' of k random noises;
 Update the generator G_ϕ by descending the gradient of (9) with weight decay;
 Update the generated samples $\mathbf{x}^{(n+j)} = G_{\phi^*}(\mathbf{z}_j)$ for $j = 1, \dots, m$ on \mathcal{Z}_m ;
end for

over generated samples, which could make $U(f_w, g_\phi^*)$ be unbounded above. This is why the WGAN must be trained by clipping the network weights of f_w on a bounded box to prevent $U(f_w, g_\phi^*)$ from becoming unbounded above.

On the contrary, the LS-GAN treats real and generated examples in pairs, and maximizes the difference of their losses up to a designated metric. Specifically, as shown in the second term of Eq. (6), when the loss of a generated sample \mathbf{z}_G becomes too large wrt that of a paired real example \mathbf{x} , the maximization of $L_\theta(\mathbf{z}_G)$ will stop if the difference $L_\theta(\mathbf{z}_G) - L_\theta(\mathbf{x})$ exceeds $\Delta(\mathbf{x}, \mathbf{z}_G)$. This prevents the minimization problem (6) unbounded below, making it well posed to solve.

More importantly, paring real and generated samples in $(\cdot)_+$ prevents their losses from being decomposed into two separate first-order moments like in the WGAN. Instead, the LS-GAN makes pairwise comparison between the losses of real and generated samples, thereby enforcing real and generated samples to coordinate with each other to learn the optimal loss function. Specifically, when a generated sample becomes close to a paired real example, the LS-GAN will stop increasing the margin $L_\theta(\mathbf{z}_G) - L_\theta(\mathbf{x})$ between their losses. As aforementioned, this allows the LS-GAN to focus on improved poor samples that are far apart from the manifold of real examples, instead of wasting its modeling capacity on those well generated samples that are already close to the manifold of real examples. This makes the LS-GAN more efficient in investing its generative ability over samples.

6 CONDITIONAL LS-GAN

The LS-GAN can easily be generalized to produce a sample based on a given condition \mathbf{y} , yielding a new paradigm of Conditional LS-GAN (CLS-GAN).

For example, if the condition is an image class, the CLS-GAN seeks to produce images of the given class; otherwise, if a text description is given as a condition, the model attempts to generate images aligned with the given description. This gives us more flexibility in controlling what samples to be generated.

Formally, the generator of CLS-GAN takes a condition vector \mathbf{y} as input along with a noise vector \mathbf{z} to produce

a sample $G_\phi(\mathbf{z}, \mathbf{y})$. To train the model, we define a loss function $L_\theta(\mathbf{x}, \mathbf{y})$ to measure the degree of the misalignment between a data sample \mathbf{x} and a given condition \mathbf{y} .

For a real example \mathbf{x} aligned with the condition \mathbf{y} , its loss function should be smaller than that of a generated sample by a margin of $\Delta(\mathbf{x}, G_\phi(\mathbf{z}, \mathbf{y}))$. This results in the following constraint,

$$L_\theta(\mathbf{x}, \mathbf{y}) \leq L_\theta(G_\phi(\mathbf{z}, \mathbf{y}), \mathbf{y}) - \Delta(\mathbf{x}, G_\phi(\mathbf{z}, \mathbf{y})) \quad (14)$$

Like the LS-GAN, this type of constraint yields the following non-zero-sum game to train the CLS-GAN, which seeks a Nash equilibrium (θ^*, ϕ^*) so that θ^* minimizes

$$\begin{aligned} S(\theta, \phi^*) = & \mathbb{E}_{\mathbf{x}, \mathbf{y} \sim P_{data}(\mathbf{x}, \mathbf{y})} L_\theta(\mathbf{x}, \mathbf{y}) \\ & + \lambda \mathbb{E}_{\substack{\mathbf{x}, \mathbf{y} \sim P_{data}(\mathbf{x}, \mathbf{y}) \\ \mathbf{z} \sim P_z(\mathbf{z})}} (\Delta(\mathbf{x}, G_{\phi^*}(\mathbf{z}, \mathbf{y})) + L_\theta(\mathbf{x}, \mathbf{y}) \\ & - L_\theta(G_{\phi^*}(\mathbf{z}, \mathbf{y}), \mathbf{y}))_+ \end{aligned} \quad (15)$$

and ϕ^* minimizes

$$T(\theta^*, \phi) = \mathbb{E}_{\substack{\mathbf{y} \sim P_{data}(\mathbf{y}) \\ \mathbf{z} \sim P_z(\mathbf{z})}} L_{\theta^*}(G_\phi(\mathbf{z}, \mathbf{y}), \mathbf{y}) \quad (16)$$

Playing the above game will lead to a trained pair of loss function L_{θ^*} and generator G_{ϕ^*} . We can show that the learned generator $G_{\phi^*}(\mathbf{z}, \mathbf{y})$ can produce samples whose distribution follows the true data density $P_{data}(\mathbf{x}|\mathbf{y})$ for a given condition \mathbf{y} .

To prove this, we say a loss function $L_\theta(\mathbf{x}, \mathbf{y})$ is Lipschitz if it is Lipschitz continuous in its first argument \mathbf{x} . We make the following assumption.

Assumption 2. For each \mathbf{y} , the conditional density $P_{data}(\mathbf{x}|\mathbf{y})$ is Lipschitz, and is supported in a convex compact set of \mathbf{x} .

Then it is not difficult to prove the following theorem, which shows that the conditional density $P_{G^*}(\mathbf{x}|\mathbf{y})$ becomes $P_{data}(\mathbf{x}|\mathbf{y})$ as $\lambda \rightarrow +\infty$. Here $P_{G^*}(\mathbf{x}|\mathbf{y})$ denotes the density of samples generated by $G_{\phi^*}(\mathbf{z}, \mathbf{y})$ with sampled random noise \mathbf{z} .

Theorem 3. Under Assumption 2, a Nash equilibrium (θ^*, ϕ^*) exists such that

- (i) $L_{\theta^*}(\mathbf{x}, \mathbf{y})$ is Lipschitz continuous in \mathbf{x} for each \mathbf{y} ;
- (ii) $P_{G^*}(\mathbf{x}|\mathbf{y})$ is Lipschitz continuous;
- (iii) $\int_{\mathbf{x}} |P_{data}(\mathbf{x}|\mathbf{y}) - P_{G^*}(\mathbf{x}|\mathbf{y})| d\mathbf{x} \leq \frac{2}{\lambda}$.

In addition, similar upper and lower bounds can be derived to characterize the learned conditional loss function $L_\theta(\mathbf{x}, \mathbf{y})$ following the same idea for LS-GAN.

A useful byproduct of the CLS-GAN is one can use the learned loss function $L_{\theta^*}(\mathbf{x}, \mathbf{y})$ to predict the label of an example \mathbf{x} by

$$\mathbf{y}^* = \arg \min_{\mathbf{y}} L_{\theta^*}(\mathbf{x}, \mathbf{y}) \quad (17)$$

The advantage of such a CLS-GAN classifier is it is trained with both labeled and generated examples, the latter of which can improving the training of the classifier by revealing more potential variations within different classes of samples. It also provides a way to evaluate the model based on its classification performance. This is an objective metric we can use to assess the quality of feature representations learned by the model.

For a classification task, a suitable value should be set to λ . Although Theorem 3 shows P_{G^*} would converge to the true conditional density P_{data} by increasing λ , it only ensures it is a good generative rather than classification model. However, a too large value of λ tends to ignore the first loss minimization term of (15) that plays an important role in minimizing classification error. Thus, a trade-off should be made to balance between classification and generation objectives.

6.1 Semi-Supervised LS-GAN

The above CLS-GAN can be considered as a fully supervised model to classify examples into different classes. It can also be extended to a Semi-Supervised model by incorporating unlabeled examples.

Suppose we have c classes indexed by $\{1, 2, \dots, c\}$. In the CLS-GAN, for each class, we choose a loss function that, for example, can be defined as the negative log-softmax,

$$L_\theta(\mathbf{x}, \mathbf{y} = l) = -\log \frac{\exp(\mathbf{a}_l(\mathbf{x}))}{\sum_{l=1}^c \exp(\mathbf{a}_l(\mathbf{x}))}$$

where $\mathbf{a}_l(\mathbf{x})$ is the l th activation output from a network layer.

Suppose we also have unlabeled examples available, and we can define a new loss function for these unlabeled examples so that they can be involved in training the CLS-GAN. Consider an unlabeled example \mathbf{x} , its groundtruth label is unknown. However, the best guess of its label can be made by choosing the one that minimizes $L_\theta(\mathbf{x}, \mathbf{y} = l)$ over l , and this inspires us to define the following loss function for the unlabeled example as

$$L_\theta^{\text{ul}}(\mathbf{x}) \triangleq \min_l L_\theta(\mathbf{x}, \mathbf{y} = l)$$

Here we modify $L_\theta(\mathbf{x}, \mathbf{y} = l)$ to $-\log \frac{\exp(\mathbf{a}_l(\mathbf{x}))}{1 + \sum_{l=1}^c \exp(\mathbf{a}_l(\mathbf{x}))}$ so $\frac{1}{1 + \sum_{l=1}^c \exp(\mathbf{a}_l(\mathbf{x}))}$ can be viewed as the probability that \mathbf{x} does not belong to any known label.

Then we have the following loss-sensitive objective that explores unlabeled examples to train the CLS-GAN,

$$\begin{aligned} S^{\text{ul}}(\theta, \phi^*) \triangleq & \mathbb{E}_{\substack{\mathbf{x} \sim P_{data}(\mathbf{x}) \\ \mathbf{z} \sim P_z(\mathbf{z})}} (\Delta(\mathbf{x}, G_{\phi^*}(\mathbf{z})) + L_\theta^{\text{ul}}(\mathbf{x}) - L_\theta^{\text{ul}}(G_{\phi^*}(\mathbf{z})))_+ \end{aligned} \quad (18)$$

This objective is combined with $S(\theta, \phi^*)$ defined in (15) to train the loss function network by minimizing

$$S(\theta, \phi^*) + \gamma S^{\text{ul}}(\theta, \phi^*)$$

where γ is a positive hyperparameter balancing the contributions from labeled and unlabeled examples.

The idea of extending the GAN for semi-supervised learning has been proposed in Salimans et al. [2], where generated samples are assigned to an artificial class, and unlabeled examples are treated as the negative examples. Our proposed semi-supervised learning differs in creating a new loss function for unlabeled examples from the losses for existing classes, by minimizing which we make the best guess of the classes of unlabeled examples. The guessed labeled will provide additional information to train the CLS-GAN model, and the updated model will in turn improve

the guess over the training course. The experiments in the following section will show that this approach can generate very competitive performance especially when the labeled data is very limited.

7 EXPERIMENTS

Objective evaluation of a data generative model is not an easy task as there is no consensus criteria to quantify the quality of generated samples. For this reason, we make a qualitative analysis of generated images, and use image classification as a surrogate to quantitatively evaluate the resultant LS-GAN model.

First, we will assess the generated images by the unconditional LS-GAN, and compare it with the other state-of-the-art GAN model. Then, we will make an objective evaluation by using the learned loss function in the CLS-GAN to classify images. This task evaluates the quality of the feature representations extracted by the CLS-GAN in terms of its classification accuracy directly. We will compare it with the feature representations extracted by the other deep generative networks.

We will also conduct a qualitative evaluation of the generated images by the CLS-GAN for different classes, and analyze the factors that would affect image generation performance. It will give us an intuitive idea of why and how the CLS-GAN can capture class-invariant feature representations to classify and generate images for various classes.

7.1 Architectures

The detail of the network architecture we adopt for training CLS-GAN on these two datasets is presented in Table 1.

Specifically, we adopt the ideas behind the network architecture for the DCGAN [7] to build the generator and the loss function networks. Compared with the conventional CNNs, maxpooling layers are replaced with strided convolutions in both networks, and fractionally-strided convolutions are used in the generator network to upsample feature maps across layers to finer resolutions. Batch-normalization layers are added in both networks between convolutional layers, and fully connected layers are removed from these networks.

However, unlike the DCGAN, the LS-GAN model (unconditional version in Section 3) does not use a sigmoid layer as the output for the loss function network. Instead, we remove it and directly output the activation before the removed sigmoid layer. This is because for a unconditional LS-GAN, we can disregard the first loss minimization term (see the remark after Lemma 1). In this case, any form of loss function can be adopted without nonnegative constraint.

On the other hand, for the loss function network in CLS-GAN, a global mean-pooling layer is added on top of convolutional layers. This produces a 1×1 feature map that is fed into a cross-entropy cost function to output the loss $L_\theta(\mathbf{x}, \mathbf{y})$ conditioned on a given class \mathbf{y} .

In the generator network, Tanh is used to produce images whose pixel values are scaled to $[-1, 1]$. Thus, all image examples in datasets are preprocessed to have their pixel values in $[-1, 1]$. More details about the design of network

TABLE 1

The Network architecture used in LS-GAN for training CIFAR-10 and SVHN, where BN stands for batch normalization, LeakyReLU for Leaky Rectifier with a slope of 0.2 for negative value, and "3c1s96o Conv." means a 3×3 convolution kernel with stride 1 and 96 outputs, while "UpConv." denotes the fractionally-stride convolution.

(a) Loss Function Network	
Input $32 \times 32 \times 3$	
3c1s96o Conv. BN LeakyReLU	
3c1s96o Conv. BN LeakyReLU	
4c2s96o Conv. BN LeakyReLU	
3c1s192o Conv. BN LeakyReLU	
3c1s192o Conv. BN LeakyReLU	
4c2s192o Conv. BN LeakyReLU	
3c1s192o Conv. BN LeakyReLU	
3c1s192o Conv. BN LeakyReLU	
1c1s192o Conv. BN LeakyReLU	
global meanpool	
Output $32 \times 32 \times 10$	
(b) Generator Network	
Input 100-D random vector + 10-D one-hot vector	
4c1s512o UpConv. BN LeakyReLU	
4c2s256o UpConv. BN LeakyReLU	
4c2s128o UpConv. BN LeakyReLU	
4c2s3o UpConv. BN LeakyReLU	
Elementwise Tanh	
Output $32 \times 32 \times 3$	

architectures can be found in literature [7]. Table 1 shows the network architecture for the CLS-GAN model on CIFAR-10 and SVHN datasets in the experiments. In particular, the architecture of the loss function network is adapted from that used in [21] with nine hidden layers.

7.2 Training Details

The models are trained in a mini-batch of 64 images, and their weights are initialized from a zero-mean Gaussian distribution with a standard deviation of 0.02. The Adam optimizer [25] is used to train the network with initial learning rate and β_1 being set to 10^{-3} and 0.5 respectively, while the learning rate is annealed every 25 epochs by a factor of 0.8.

The hyperparameter γ and λ are chosen via a five-fold cross-validation. We also test different types of distance functions for $\Delta(\cdot, \cdot)$ in the model, and find pixel-wise L_1 distance achieves better performance than the other compared choices like L_2 distance. Thus we choose L_1 distance through the experiments. We also tried intermediate feature maps from the loss function network to compute the distance between images as the margin. But we find the results are not as good as the L_1 distance. We found that these intermediate feature maps would collapse to a trivial single point as the margin $\Delta(\cdot, \cdot)$ reduces to zero through training.

For the generator network of LS-GAN, it tasks a 100-dimensional random vector drawn from $\text{Unif}[-1, 1]$ as input. For the CLS-GAN generator, an one-hot vector encoding the image class is concatenated with the sampled

random vector as its input. We will train CLS-GAN as presented in Section 6 by involving both unlabeled and labeled examples. This will be compared against the other state-of-the-art supervised deep generative models as well as the other GAN models in literature.

7.3 Generated Images by LS-GAN

First we make a qualitative comparison between the images generated by the DCGAN and the LS-GAN on the celebA dataset.

As illustrated in Figure 3, there is no perceptible difference between the qualities of generated images by two compared GAN models after they are trained after 25 epochs.

It is known that the DCGAN architecture has been optimized for the classic GAN training criterion to reach the maximal performance. It is susceptible that this architecture could be fragile if we change some of its part. So we wish to test if the LS-GAN can be more robust than the DCGAN when a change is made to the network architecture.

For example, one of most important component is the batch normalization inserted between the fractional convolution layers in the generator network. It has been reported in literature [2] that the batch normalization not only plays a key role in training a satisfactory DCGAN model, but also prevents the collapse of the model into few modes of data density.

As illustrated in Figure 4, if one removed the batch normalization layers from the generator, the DCGAN would collapse without generating meaningful face images. On the contrary, the LS-GAN still performs very well even if these batch normalization layers are removed, and there is **no perceived deterioration or mode collapse of its generated images**. This shows that the LS-GAN can be much robust to the architecture changes.

We also illustrate the norm of the generator’s gradient (in logarithmic scale) over iterations in Figure 5. The generator is updated only every 1, 3, and 5 iterations while the loss function is updated every iteration. Since the loss function can be quickly updated to be optimal, the figure shows the generator’s gradient does not vanish even if the loss function becomes well trained.

From the figure, we note that the norm of the generator’s gradient, no matter how more frequently the loss function is updated than the generator, gradually increase over iterations, until it stops at a constant level. This suggests the generator’s cost tends to become roughly linear rather than being saturated through the training procedure. Thus, it can provide sufficient gradient to continuously update the generator, making it more amenable to train the model.

7.4 Image Classification

We conduct experiments on CIFAR-10 and SVHN to compare the classification accuracy of LS-GAN with the other approaches.

7.4.1 CIFAR-10

The CIFAR dataset [30] consists of 50,000 training images and 10,000 test images on ten image categories. We test the proposed CLS-GAN model with class labels as conditions. In the supervised training, all labeled examples are used to

train the CLS-GAN. The experiment results on this task are reported by averaging over ten subsets of labeled examples.

We also conduct experiments with 400 labeled examples per class, which is a more challenging task as much fewer labeled examples are used for training. In this case, the remaining unlabeled examples are used to train the model as well in a semi-supervised fashion as discussed in Section 6. In each mini-batch, the same number of labeled and unlabeled examples are used to update the model by stochastic gradient descent.

Both hyperparameters γ and λ are chosen via a five-fold cross-validation on the labeled examples from $\{0.25, 0.5, 1.0, 2.0\}$ and $\{0.5, 1.0, 2.0\}$ respectively. Once they are chosen, the model is trained with the chosen hyperparameters on the whole training set, and the performance is reported based on the results on the test set.

We compare the proposed model with the state-of-the-art methods in literature. In particular, we compare with the conditional GAN [29] as well as the DCGAN [7]. For the sake of fair comparison, the conditional GAN shares the same architecture as the CLS-GAN. On the other hand, the DCGAN algorithm [7] max-pools the discriminator’s convolution features from all layers to 4×4 grids as the image features, and a L2-SVM is then trained to classify images. The DCGAN is an unsupervised model which has shown competitive performance on generating photo-realistic images. Its feature representations are believed to reach the state-of-the-art performance in modeling images with no supervision.

We also compare with the other recently developed supervised and semi-supervised models in literature, including the baseline 1 Layer K-means feature extraction pipeline, a multi-layer extension of the baseline model (3 Layer K-means Learned RF [26]), View Invariant K-means [27], Exemplar CNN [28], Ladder Network [19], as well as CatGAN [21] and improved GAN [2]. In particular, among the compared semi-supervised algorithms, the improved GAN has recorded the best performance in literature. The CLS-GAN adopts the same network architecture used in the improved GAN to make a direct comparison.

Table 2 compares the experiment results, showing the CLS-GAN successfully outperforms the compared algorithms in both fully-supervised and semi-supervised settings.

7.4.2 SVHN

The SVHN (i.e., Street View House Number) dataset [31] contains 32×32 color images of house numbers collected by Google Street View. They are roughly centered on a digit in a house number, and the objective is to recognize the digit. The training set has 73,257 digits while the test set consists of 26,032.

To test the model, 1,000 labeled digits are used to train the model, which are uniformly selected from ten digit classes, that is 100 labeled examples per digit class. The remaining unlabeled examples are used as additional data to enhance the generative ability of CLS-GAN in semi-supervised fashion. We expect a good generative model could produce additional examples to augment the training set.

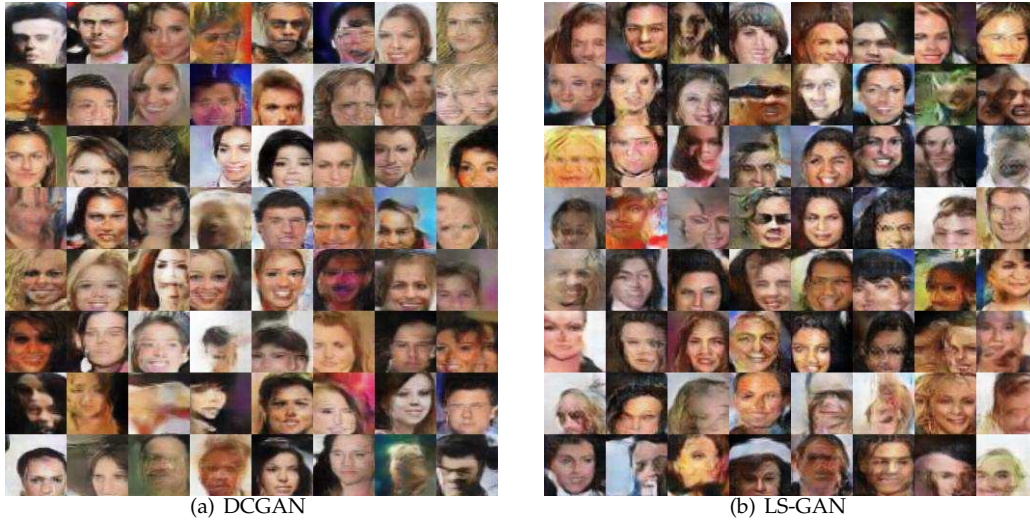


Fig. 3. Images generated by the DCGAN and the LS-GAN on the CelebA dataset. The results are obtained after 25 epochs of training the models.

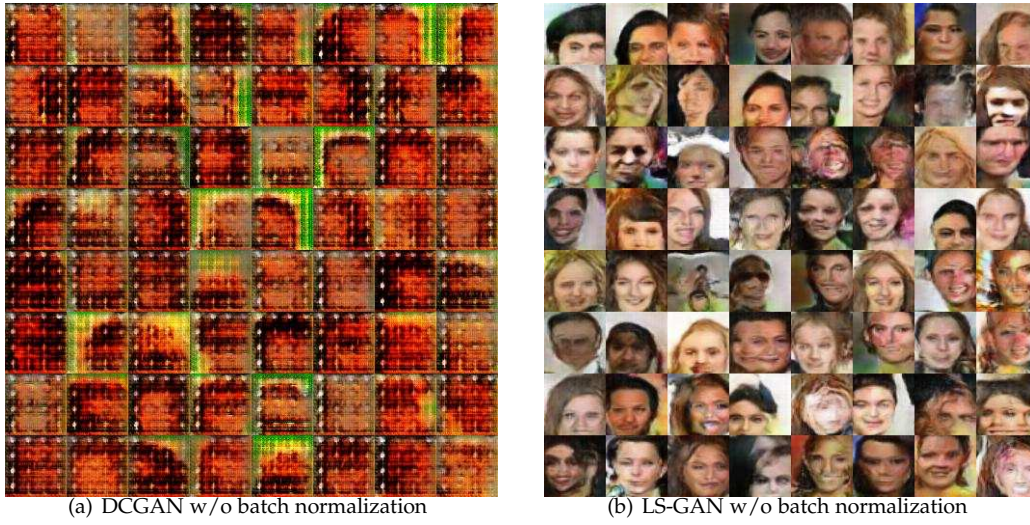


Fig. 4. Images generated by the DCGAN and the LS-GAN on the CelebA dataset without batch normalization for the generator networks. The results are obtained after 25 epochs of training the models.

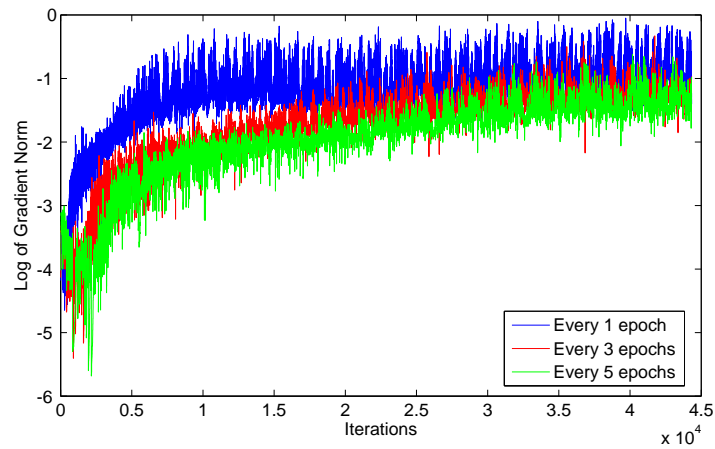


Fig. 5. The log of the generator's gradient norm over iterations. The generator is updated every 1, 3, and 5 iterations while the loss function is updated every iteration. The loss function can be quickly updated to be optimal, and the figure shows the generator's gradient does not vanish even if the loss function is well trained.

TABLE 2

Classification results on CIFAR-10 dataset, in comparison with the state-of-the-art methods. Both accuracies with all training examples labeled (all) and only 400 labeled examples per class (400) are reported. The best result is highlight in bold. For those semi-supervised learning algorithms, the accuracy (all) is not reported (–) as all training examples are labeled.

<i>Methods</i>	<i>Accuracy (All)</i>	<i>Accuracy (400 per class)</i>
1 Layer K-means [7]	80.6%	63.7% ($\pm 0.7\%$)
3 Layer K-means Learned RF [26]	82.0%	70.7% ($\pm 0.7\%$)
View Invariant K-means [27]	81.9%	72.6% ($\pm 0.7\%$)
Exemplar CNN [28]	84.3%	77.4% ($\pm 0.2\%$)
Conditional GAN [29]	83.6%	75.5% ($\pm 0.4\%$)
DCGAN [7]	82.8%	73.8% ($\pm 0.4\%$)
Ladder Network [19]	–	79.6% ($\pm 0.5\%$)
CatGAN [21]	–	80.4% ($\pm 0.4\%$)
Improved GAN [2]	–	81.4% ($\pm 2.3\%$)
CLS-GAN	91.7%	82.7% ($\pm 0.5\%$)

TABLE 3

Classification results on SVHN dataset with 1,000 labeled examples. The best result is highlighted in bold.

<i>Methods</i>	<i>Error rate</i>
KNN [7]	77.93%
TSVM [7]	66.55%
M1+KNN [17]	65.63%
M1+TSVM [17]	54.33%
M1+M2 [17]	36.02%
SWWAE w/o dropout [32]	27.83%
SWWAE with dropout [32]	23.56%
DCGAN [7]	22.48%
Conditional GAN [29]	21.85% $\pm 0.38\%$
Supervised CNN [7]	28.87%
DGN [17]	36.02% $\pm 0.10\%$
Virtual Adversarial [33]	24.63%
Auxiliary DGN [34]	22.86%
Skip DGN [34]	16.61% $\pm 0.24\%$
Improved GAN [2]	8.11% $\pm 1.3\%$
CLS-GAN	5.98%$\pm 0.27\%$

We use the same experiment setup and network architecture for CIFAR-10 dataset. Table 3 reports the results on the SVHN dataset. It shows that LS-GAN outperforms the compared algorithms again.

7.4.3 Analysis of Generated Images by CLS-GAN

Figure 6 illustrates the generated images by CLS-GAN for MNIST, CIFAR-10 and SVHN datasets. On each dataset, images in a column are generated for the same class. On the MNIST and the SVHN, both handwritten and street-view digits are quite legible. Both also cover many variants for each digit class. For example, the synthesized MNIST digits have various writing styles, rotations and sizes, and the generated SVHN digits have various lighting conditions, sizes and even different co-occurring digits in the cropped bounding boxes. On the CIFAR-10 dataset, image classes can be recognized from the generated images although some visual details are missing. This is because the images in the CIFAR-10 dataset have very low resolution (32×32 pixels), and most details are even missing from input examples.

We also observe that if we set a small value to the hyperparameter λ , the generated images would become

very similar to each other within each class. As illustrated in Figure 7, the images are generated by halving λ used for generating images in Figure 6. A smaller λ means a relatively large weight is placed on the first loss minimization term of (8), which tends to collapse generated images to a single mode as it aggressively minimizes their losses to train the generator. This is also consistent with Theorem 3 where the density of generated samples with a smaller λ could have a larger deviation from the underlying density. One should avoid the collapse of trained generator since diversifying generated images can improve the classification performance of the CLS-GAN by revealing more intra-class variations. This will help improve the model’s generalization ability as these variations could appear in future images.

However, one should also avoid setting too large value to λ . Otherwise, the role of the first loss minimization term could be underestimated, which can also adversely affect the classification results without reducing the training loss to a satisfactory level. Therefore, we choose a proper value for λ by cross-validation on the training set in the experiments.

In brief, the comparison between Figure 6 and Figure 7 reveals a trade-off between image generation quality and classification accuracy through the hyperparameter λ . Such a trade-off is intuitive: while a classification task usually focuses on learning class-invariant representations that do not change within a class, image generation should be able to capture many variant factors (e.g., lighting conditions, viewing angles, and object poses) so that it could diversify generated samples for each class. Although diversified examples can augment training dataset, it comes at a cost of trading class-invariance for modeling variant generation factors. Perhaps, this is an intrinsic dilemma between supervised learning and data generation that is worth more theoretical and empirical studies in future.

8 CONCLUSION

In this paper, we present a novel Loss-Sensitive GAN (LS-GAN) approach to generate samples from the underlying distributions. The LS-GAN learns a loss function to distinguish generated samples from given examples by imposing the constraint that the loss of a real sample should be



Fig. 6. Images generated by CLS-GAN for MNIST, CIFAR-10 and SVHN datasets. For each dataset, images in a column are generated according to the same class. In particular, the generated CIFAR-10 images are airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck from the leftmost to the rightmost column.

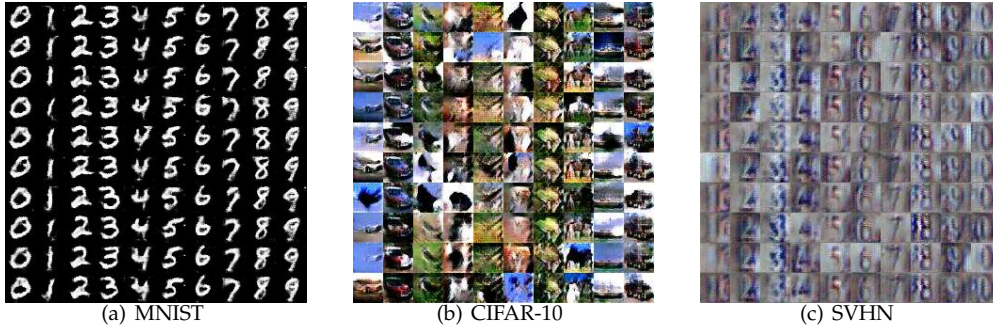


Fig. 7. Illustration of generated images that are similar to one another as they are collapsed to a single modes of the underlying image density on MNIST, CIFAR-10 and SVHN datasets.

sufficiently small by a data dependant margin than that of a generated sample. As the data generator improves, the gap between the losses of real and generated samples will be gradually closed when the generated samples become better and better.

We prove that this approach can generate samples whose data density matches the underlying true data density. Our analysis also suggests that we do not need to assume the model has infinite modeling capacity to obtain this result. Instead a class of generators and loss functions with bounded Lipschitz constants are sufficient to show the consistency of generated samples with the underlying data density. A non-parametric solution is derived to characterize the lower and upper bounds of the learned loss functions sought by the LS-GAN. We show both bounds are piecewise linear with non-vanishing gradient, which sheds some light on how the LS-GAN generator can be sufficiently updated by the loss function even if it is trained to be optimal.

Moreover, we generalize the LS-GAN to a Conditional LS-GAN (CLS-GAN) that can generate samples under given conditions. Similar theoretical guarantee can be established on the CLS-GAN so that the conditional density of its generated samples is consistent with the true conditional density. Experiment results demonstrate the competitive image classification accuracy by using the learned loss function, as well as the high quality of generated images.

REFERENCES

- [1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.
- [2] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in Neural Information Processing Systems*, 2016, pp. 2226–2234.
- [3] M. Saito and E. Matsumoto, "Temporal generative adversarial nets," *arXiv preprint arXiv:1611.06624*, 2016.
- [4] O. Mogren, "C-rnn-gan: Continuous recurrent neural networks with adversarial training," *arXiv preprint arXiv:1611.09904*, 2016.
- [5] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: sequence generative adversarial nets with policy gradient," *arXiv preprint arXiv:1609.05473*, 2016.
- [6] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks,"
- [7] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.
- [8] D. H. Wolpert, "The lack of a priori distinctions between learning algorithms," *Neural computation*, vol. 8, no. 7, pp. 1341–1390, 1996.
- [9] E. L. Denton, S. Chintala, R. Fergus *et al.*, "Deep generative image models using a laplacian pyramid of adversarial networks," in *Advances in neural information processing systems*, 2015, pp. 1486–1494.
- [10] D. J. Im, C. D. Kim, H. Jiang, and R. Memisevic, "Generating images with recurrent adversarial networks," *arXiv preprint arXiv:1602.05110*, 2016.
- [11] J. Zhao, M. Mathieu, and Y. LeCun, "Energy-based generative adversarial network," *arXiv preprint arXiv:1609.03126*, 2016.
- [12] S. Nowozin, B. Cseke, and R. Tomioka, "f-gan: Training generative neural samplers using variational divergence minimization," *arXiv preprint arXiv:1606.00709*, 2016.
- [13] X. Chen, Y. Duan, R. Houthoofd, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2016, pp. 2172–2180.
- [14] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," *arXiv preprint arXiv:1508.06576*, 2015.
- [15] A. Dosovitskiy, J. Tobias Springenberg, and T. Brox, "Learning to generate chairs with convolutional neural networks," in *Pro-*

ceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1538–1546.

- [16] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra, “Draw: A recurrent neural network for image generation,” *arXiv preprint arXiv:1502.04623*, 2015.
- [17] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, “Semi-supervised learning with deep generative models,” in *Advances in Neural Information Processing Systems*, 2014, pp. 3581–3589.
- [18] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *arXiv preprint arXiv:1312.6114*, 2013.
- [19] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko, “Semi-supervised learning with ladder networks,” in *Advances in Neural Information Processing Systems*, 2015, pp. 3546–3554.
- [20] H. Valpola, “From neural pca to deep unsupervised learning,” *Adv. in Independent Component Analysis and Learning Machines*, pp. 143–171, 2015.
- [21] J. T. Springenberg, “Unsupervised and semi-supervised learning with categorical generative adversarial networks,” *arXiv preprint arXiv:1511.06390*, 2015.
- [22] K. C. Border, *Fixed point theorems with applications to economics and game theory*. Cambridge university press, 1989.
- [23] D. Carando, R. Fraiman, and P. Groisman, “Nonparametric likelihood based estimation for a multivariate lipschitz density,” *Journal of Multivariate Analysis*, vol. 100, no. 5, pp. 981–992, 2009.
- [24] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein gan,” *arXiv preprint arXiv:1701.07875*, January 2017.
- [25] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [26] A. Coates and A. Y. Ng, “Selecting receptive fields in deep networks,” in *Advances in Neural Information Processing Systems*, 2011, pp. 2528–2536.
- [27] K. Y. Hui, “Direct modeling of complex invariances for visual object features,” in *International Conference on Machine Learning*, 2013, pp. 352–360.
- [28] A. Dosovitskiy, P. Fischer, J. T. Springenberg, M. Riedmiller, and T. Brox, “Discriminative unsupervised feature learning with exemplar convolutional neural networks.”
- [29] M. Mirza and S. Osindero, “Conditional generative adversarial nets,” *arXiv preprint arXiv:1411.1784*, 2014.
- [30] A. Krizhevsky, “Learning multiple layers of features from tiny images,” 2009.
- [31] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, “Reading digits in natural images with unsupervised feature learning,” 2011.
- [32] J. Zhao, M. Mathieu, R. Goroshin, and Y. Lecun, “Stacked what-where auto-encoders,” *arXiv preprint arXiv:1506.02351*, 2015.
- [33] T. Miyato, S.-i. Maeda, M. Koyama, K. Nakae, and S. Ishii, “Distributional smoothing by virtual adversarial examples,” *arXiv preprint arXiv:1507.00677*, 2015.
- [34] L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther, “Auxiliary deep generative models,” *arXiv preprint arXiv:1602.05473*, 2016.

APPENDIX A PROOF OF LEMMA 1

To prove Lemma 1, we need the following lemma.

Lemma 4. *For two probability densities $p(\mathbf{x})$ and $q(\mathbf{x})$, if $p(\mathbf{x}) \geq \eta q(\mathbf{x})$ almost everywhere, we have*

$$\int_{\mathbf{x}} |p(\mathbf{x}) - q(\mathbf{x})| d\mathbf{x} \leq \frac{2(1-\eta)}{\eta}$$

for $\eta \in (0, 1]$.

Proof. We have the following equalities and inequalities:

$$\begin{aligned} & \int_{\mathbf{x}} |p(\mathbf{x}) - q(\mathbf{x})| d\mathbf{x} \\ &= \int_{\mathbf{x}} \mathbb{1}_{[p(\mathbf{x}) \geq q(\mathbf{x})]} (p(\mathbf{x}) - q(\mathbf{x})) d\mathbf{x} \\ &+ \int_{\mathbf{x}} \mathbb{1}_{[p(\mathbf{x}) < q(\mathbf{x})]} (q(\mathbf{x}) - p(\mathbf{x})) d\mathbf{x} \\ &= \int_{\mathbf{x}} (1 - \mathbb{1}_{[p(\mathbf{x}) < q(\mathbf{x})]}) (p(\mathbf{x}) - q(\mathbf{x})) d\mathbf{x} \\ &+ \int_{\mathbf{x}} \mathbb{1}_{[p(\mathbf{x}) < q(\mathbf{x})]} (q(\mathbf{x}) - p(\mathbf{x})) d\mathbf{x} \\ &= 2 \int_{\mathbf{x}} \mathbb{1}_{[p(\mathbf{x}) < q(\mathbf{x})]} (q(\mathbf{x}) - p(\mathbf{x})) d\mathbf{x} \\ &\leq 2 \left(\frac{1}{\eta} - 1 \right) \int_{\mathbf{x}} \mathbb{1}_{[p(\mathbf{x}) < q(\mathbf{x})]} p(\mathbf{x}) d\mathbf{x} \\ &\leq \frac{2(1-\eta)}{\eta} \end{aligned} \tag{19}$$

This completes the proof. \square

Now we can prove Lemma 1.

Proof. Suppose (θ^*, ϕ^*) is a Nash equilibrium for the problem (6) and (7).

Then, on one hand, we have

$$\begin{aligned} S(\theta^*, \phi^*) &\geq \mathbb{E}_{\mathbf{x} \sim P_{data}(\mathbf{x})} L_{\theta^*}(\mathbf{x}) \\ &+ \lambda \mathbb{E}_{\substack{\mathbf{x} \sim P_{data}(\mathbf{x}) \\ \mathbf{z}_G \sim P_{G^*}(\mathbf{z}_G)}} (\Delta(\mathbf{x}, \mathbf{z}_G) + L_{\theta^*}(\mathbf{x}) - L_{\theta^*}(\mathbf{z}_G)) \\ &= \int_{\mathbf{x}} P_{data}(\mathbf{x}) L_{\theta^*}(\mathbf{x}) d\mathbf{x} + \lambda \mathbb{E}_{\substack{\mathbf{x} \sim P_{data}(\mathbf{x}) \\ \mathbf{z}_G \sim P_{G^*}(\mathbf{z}_G)}} \Delta(\mathbf{x}, \mathbf{z}_G) \\ &+ \lambda \int_{\mathbf{x}} P_{data}(\mathbf{x}) L_{\theta^*}(\mathbf{x}) d\mathbf{x} - \lambda \int_{\mathbf{z}_G} P_{G^*}(\mathbf{z}_G) L_{\theta^*}(\mathbf{z}_G) d\mathbf{z}_G \\ &= \int_{\mathbf{x}} ((1 + \lambda) P_{data}(\mathbf{x}) - \lambda P_{G^*}(\mathbf{x})) L_{\theta^*}(\mathbf{x}) d\mathbf{x} \\ &+ \lambda \mathbb{E}_{\substack{\mathbf{x} \sim P_{data}(\mathbf{x}) \\ \mathbf{z}_G \sim P_{G^*}(\mathbf{z}_G)}} \Delta(\mathbf{x}, \mathbf{z}_G) \end{aligned} \tag{20}$$

where the first inequality follows from $(a)_+ \geq a$.

We also have $T(\theta^*, \phi^*) \leq T(\theta^*, \phi)$ for any G_ϕ as ϕ^* minimizes $T(\theta^*, \phi)$. In particular, we can replace $P_G(\mathbf{x})$ in $T(\theta^*, \phi)$ with $P_{data}(\mathbf{x})$, which yields

$$\int_{\mathbf{x}} L_{\theta^*}(\mathbf{x}) P_{G^*}(\mathbf{x}) d\mathbf{x} \leq \int_{\mathbf{x}} L_{\theta^*}(\mathbf{x}) P_{data}(\mathbf{x}) d\mathbf{x}.$$

Applying this inequality into (20) leads to

$$\begin{aligned} & S(\theta^*, \phi^*) \\ &\geq \int_{\mathbf{x}} P_{data}(\mathbf{x}) L_{\theta^*}(\mathbf{x}) d\mathbf{x} + \lambda \mathbb{E}_{\substack{\mathbf{x} \sim P_{data}(\mathbf{x}) \\ \mathbf{z}_G \sim P_{G^*}(\mathbf{z}_G)}} \Delta(\mathbf{x}, \mathbf{z}_G) \\ &\geq \lambda \mathbb{E}_{\substack{\mathbf{x} \sim P_{data}(\mathbf{x}) \\ \mathbf{z}_G \sim P_{G^*}(\mathbf{z}_G)}} \Delta(\mathbf{x}, \mathbf{z}_G) \end{aligned} \tag{21}$$

where the last inequality follows as $L_{\theta}(\mathbf{x})$ is nonnegative.

On the other hand, consider a particular loss function

$$L_{\theta}(\mathbf{x}) = \alpha (-(1 + \lambda) P_{data}(\mathbf{x}) + \lambda P_{G^*}(\mathbf{x}))_+ \tag{22}$$

When α is a sufficiently small positive coefficient, $L_\theta(\mathbf{x})$ is a nonexpansive function (i.e., a function with Lipschitz constant no larger than 1.). This follows from the assumption that P_{data} and P_G are Lipschitz. In this case, we have

$$\Delta(\mathbf{x}, \mathbf{z}_G) + L_\theta(\mathbf{x}) - L_\theta(\mathbf{z}_G) \geq 0 \quad (23)$$

By placing this $L_\theta(\mathbf{x})$ into $S(\theta, \phi^*)$, one can show that

$$\begin{aligned} S(\theta, \phi^*) &= \int_{\mathbf{x}} ((1 + \lambda)P_{data}(\mathbf{x}) - \lambda P_{G^*}(\mathbf{x})) L_\theta(\mathbf{x}) d\mathbf{x} \\ &+ \lambda \mathbb{E}_{\substack{\mathbf{x} \sim P_{data}(\mathbf{x}) \\ \mathbf{z}_G \sim P_{G^*}(\mathbf{z}_G)}} \Delta(\mathbf{x}, \mathbf{z}_G) \\ &= -\alpha \int_{\mathbf{x}} (-(1 + \lambda)P_{data}(\mathbf{x}) + \lambda P_{G^*}(\mathbf{x}))_+^2 d\mathbf{x} \\ &+ \lambda \mathbb{E}_{\substack{\mathbf{x} \sim P_{data}(\mathbf{x}) \\ \mathbf{z}_G \sim P_{G^*}(\mathbf{z}_G)}} \Delta(\mathbf{x}, \mathbf{z}_G) \end{aligned}$$

where the first equality uses Eq. (23), and the second equality is obtained by substituting $L_\theta(\mathbf{x})$ in Eq. (22) into the equation.

Assuming that $(1 + \lambda)P_{data}(\mathbf{x}) - \lambda P_{G^*}(\mathbf{x}) < 0$ on a set of nonzero measure, the above equation would be strictly upper bounded by $\lambda \mathbb{E}_{\substack{\mathbf{x} \sim P_{data}(\mathbf{x}) \\ \mathbf{z}_G \sim P_{G^*}(\mathbf{z}_G)}} \Delta(\mathbf{x}, \mathbf{z}_G)$ and we have

$$S(\theta^*, \phi^*) \leq S(\theta, \phi^*) < \lambda \mathbb{E}_{\substack{\mathbf{x} \sim P_{data}(\mathbf{x}) \\ \mathbf{z}_G \sim P_{G^*}(\mathbf{z}_G)}} \Delta(\mathbf{x}, \mathbf{z}_G) \quad (24)$$

This results in a contradiction with Eq. (21). Therefore, we must have

$$P_{data}(\mathbf{x}) \geq \frac{\lambda}{1 + \lambda} P_{G^*}(\mathbf{x}) \quad (25)$$

for almost everywhere. By Lemma 4, we have

$$\int_{\mathbf{x}} |P_{data}(\mathbf{x}) - P_{G^*}(\mathbf{x})| d\mathbf{x} \leq \frac{2}{\lambda}$$

Let $\lambda \rightarrow +\infty$, this leads to

$$\int_{\mathbf{x}} |P_{data}(\mathbf{x}) - P_{G^*}(\mathbf{x})| d\mathbf{x} \rightarrow 0$$

This proves that $P_{G^*}(\mathbf{x})$ converges to $P_{data}(\mathbf{x})$ as $\lambda \rightarrow +\infty$. \square

APPENDIX B

PROOF OF THEOREM 2 AND COROLLARY 2

We prove Theorem 2 as follows.

Proof. First, the existence of a minimizer follows from the fact that the functions in \mathcal{F}_κ form a compact set, and the objective function is convex.

To prove the minimizer has the two forms in (10), for each $L_\theta \in \mathcal{F}_\kappa$, let us consider

$$\begin{aligned} \hat{L}_\theta(\mathbf{x}) &= \max_{1 \leq i \leq n+m} \{ (L_\theta(\mathbf{x}^{(i)}) - \kappa \Delta(\mathbf{x}, \mathbf{x}^{(i)}))_+ \}, \\ \tilde{L}_\theta(\mathbf{x}) &= \min_{1 \leq i \leq n+m} \{ L_\theta(\mathbf{x}^{(i)}) + \kappa \Delta(\mathbf{x}, \mathbf{x}^{(i)}) \} \end{aligned}$$

It is not hard to verify that $\hat{L}_\theta(\mathbf{x}^{(i)}) = L_\theta(\mathbf{x}^{(i)})$ and $\tilde{L}_\theta(\mathbf{x}^{(i)}) = L_\theta(\mathbf{x}^{(i)})$ for $1 \leq i \leq n + m$, by noting that L_θ has its Lipschitz constant bounded by κ .

Actually, by Lipschitz continuity, we have $L_\theta(\mathbf{x}^{(i)}) - L_\theta(\mathbf{x}^{(j)}) \leq \kappa \Delta(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$, and thus

$$L_\theta(\mathbf{x}^{(j)}) - \kappa \Delta(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \leq L_\theta(\mathbf{x}^{(i)})$$

Because $L_\theta(\mathbf{x}^{(i)}) \geq 0$ by the assumption (i.e., it is lower bounded by zero), it can be shown that for all j

$$(L_\theta(\mathbf{x}^{(j)}) - \kappa \Delta(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}))_+ \leq L_\theta(\mathbf{x}^{(i)}).$$

Hence, by the definition of $\hat{L}_\theta(\mathbf{x})$ and taking the maximum over j on the left hand side, we have

$$\hat{L}_\theta(\mathbf{x}^{(i)}) \leq L_\theta(\mathbf{x}^{(i)})$$

On the other hand, we have

$$\hat{L}_\theta(\mathbf{x}^{(i)}) \geq L_\theta(\mathbf{x}^{(i)})$$

because $\hat{L}_\theta(\mathbf{x}) \geq (L_\theta(\mathbf{x}^{(i)}) - \kappa \Delta(\mathbf{x}, \mathbf{x}^{(i)}))_+$ for any \mathbf{x} , and it is true in particular for $\mathbf{x} = \mathbf{x}^{(i)}$. This shows $\hat{L}_\theta(\mathbf{x}^{(i)}) = L_\theta(\mathbf{x}^{(i)})$. Likewise, we can prove that $\tilde{L}_\theta(\mathbf{x}^{(i)}) = L_\theta(\mathbf{x}^{(i)})$.

Similarly, one can prove $\tilde{L}_\theta(\mathbf{x}^{(i)}) = L_\theta(\mathbf{x}^{(i)})$. To show this, we have

$$L_\theta(\mathbf{x}^{(j)}) + \kappa \Delta(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \geq L_\theta(\mathbf{x}^{(i)})$$

by the Lipschitz continuity of L_θ . By taking the minimum over j , we have

$$\tilde{L}_\theta(\mathbf{x}^{(i)}) \geq L_\theta(\mathbf{x}^{(i)}).$$

On the other hand, we have $\tilde{L}_\theta(\mathbf{x}^{(i)}) \leq L_\theta(\mathbf{x}^{(i)})$ by the definition of $\tilde{L}_\theta(\mathbf{x}^{(i)})$. Combining these two inequalities shows that $\tilde{L}_\theta(\mathbf{x}^{(i)}) = L_\theta(\mathbf{x}^{(i)})$.

Now we can prove for any function $L_\theta \in \mathcal{F}_\kappa$, there exist \hat{L}_θ and \tilde{L}_θ both of which attain the same value of $S_{n,m}$ as L_θ , since $S_{n,m}$ only depends on the values of L_θ on the data points $\{\mathbf{x}^{(i)}\}$. In particular, this shows that any global minimum in \mathcal{F}_κ of $S_{n,m}$ can also be attained by the corresponding functions of the form (10). By setting $l_i^* = \hat{L}_{\theta^*}(\mathbf{x}^{(i)}) = \tilde{L}_{\theta^*}(\mathbf{x}^{(i)})$ for $i = 1, \dots, n + m$, this completes the proof. \square

Finally, we prove Corollary 2 that bounds L_θ with $\hat{L}_\theta(\mathbf{x})$ and $\tilde{L}_\theta(\mathbf{x})$ constructed above.

Proof. By the Lipschitz continuity, we have

$$L_\theta(\mathbf{x}^{(i)}) - \kappa \Delta(\mathbf{x}, \mathbf{x}^{(i)}) \leq L_\theta(\mathbf{x})$$

Since $L_\theta(\mathbf{x}) \geq 0$, it follows that

$$(L_\theta(\mathbf{x}^{(i)}) - \kappa \Delta(\mathbf{x}, \mathbf{x}^{(i)}))_+ \leq L_\theta(\mathbf{x})$$

Taking the maximum over i on the left hand side, we obtain

$$\hat{L}_\theta(\mathbf{x}) \leq L_\theta(\mathbf{x})$$

This proves the lower bound.

Similarly, we have by Lipschitz continuity

$$L_\theta(\mathbf{x}) \leq \kappa \Delta(\mathbf{x}, \mathbf{x}^{(i)}) + L_\theta(\mathbf{x}^{(i)})$$

which, by taking the minimum over i on the left hand side, leads to

$$\tilde{L}_\theta(\mathbf{x}) \leq L_\theta(\mathbf{x})$$

This shows the upper bound. \square

APPENDIX C

PROOF OF LEMMA 3

Proof. Suppose a pair of (f_w^*, g_ϕ^*) jointly solve the WGAN problem.

Then, on one hand, we have

$$U(f_w^*, g_\phi^*) = \int_{\mathbf{x}} f_w^*(\mathbf{x}) P_{data}(\mathbf{x}) d\mathbf{x} - \int_{\mathbf{x}} f_w^*(\mathbf{x}) P_{g_\phi^*}(\mathbf{x}) d\mathbf{x} \leq 0 \quad (26)$$

where the inequality follows from $V(f_w^*, g_\phi^*) \geq V(f_w^*, g_\phi)$ by replacing $P_{g_\phi}(\mathbf{x})$ with $P_{data}(\mathbf{x})$.

Consider a particular $f_w(\mathbf{x}) \triangleq \alpha(P_{data}(\mathbf{x}) - P_{g_\phi^*}(\mathbf{x}))_+$. Since $P_{data}(\mathbf{x})$ and $P_{g_\phi^*}$ are Lipschitz by assumption, when α is sufficiently small, it can be shown that $f_w(\mathbf{x}) \in \mathcal{L}_1$.

Substituting this f_w into $U(f_w, g_\phi^*)$, we get

$$U(f_w, g_\phi^*) = \alpha \int_{\mathbf{x}} (P_{data}(\mathbf{x}) - P_{g_\phi^*}(\mathbf{x}))_+^2 d\mathbf{x}$$

Let us assume $P_{data}(\mathbf{x}) > P_{g_\phi^*}(\mathbf{x})$ on a set of nonzero measure, we would have

$$U(f_w, g_\phi^*) \geq U(f_w, g_\phi) > 0$$

This leads to a contradiction with (26), so we must have

$$P_{data}(\mathbf{x}) \leq P_{g_\phi^*}(\mathbf{x})$$

almost everywhere.

Hence, by Lemma 4, we prove the conclusion that

$$\int_{\mathbf{x}} |P_{data}(\mathbf{x}) - P_{g_\phi^*}(\mathbf{x})| d\mathbf{x} = 0.$$

□