Least Squares Generative Adversarial Networks

Xudong Mao[1] Qing Li[1] Haoran Xie[2]
Raymond Y.K. Lau[3] Zhen Wang[4] Stephen Paul Smolley[5]
[1]Department of Computer Science, City University of Hong Kong
[2]Department of Mathematics and Information Technology, The Education University of Hong Kong
[3]Department of Information Systems, City University of Hong Kong
[4]Center for Optical Imagery Analysis and Learning, Northwestern Polytechnical University
[5]CodeHatch Corp.
xudonmao@gmail.com, itqli@cityu.edu.hk, hrxie2@gmail.com
raylau@cityu.edu.hk, zhenwang0@gmail.com, steve@codehatch.com

Abstract

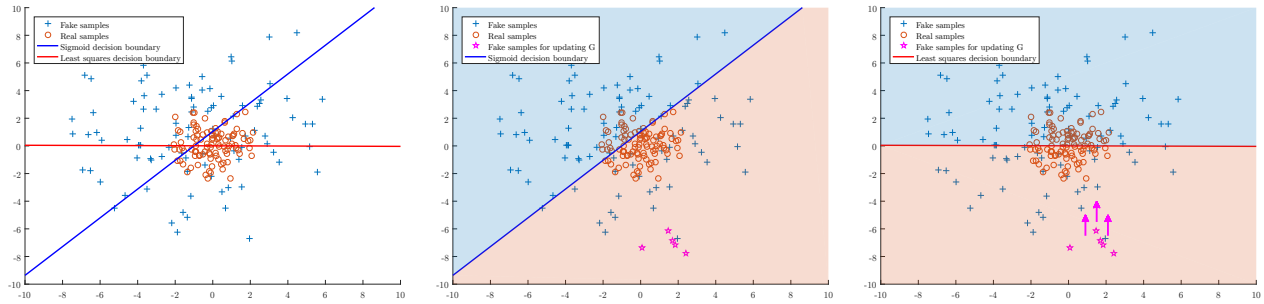
Unsupervised learning with generative adversarial networks (GANs) has proven hugely successful. Regular GANs hypothesize the discriminator as a classifier with the sigmoid cross entropy loss function. However, we found that this loss function may lead to the vanishing gradients problem during the learning process. To overcome such a problem, we propose in this paper the Least Squares Generative Adversarial Networks (LSGANs) which adopt the least squares loss function for the discriminator. We show that minimizing the objective function of LSGAN yields minimizing the Pearson χ^2 divergence. There are two benefits of LSGANs over regular GANs. First, LSGANs are able to generate higher quality images than regular GANs. Second, LSGANs perform more stable during the learning process. We evaluate LSGANs on LSUN and CIFAR-10 datasets and the experimental results show that the images generated by LSGANs are of better quality than the ones generated by regular GANs. We also conduct two comparison experiments between LSGANs and regular GANs to illustrate the stability of LSGANs.

1. Introduction

Deep learning has launched a profound reformation and even been applied to many real-world tasks, such as image classification [7], object detection [27] and segmentation [18]. These tasks obviously fall into the scope of supervised learning, which means that a lot of labeled data are provided for the learning processes. Compared with supervised learning, however, unsupervised learning tasks, such as generative models, obtain limited impact from deep learning. Although some deep generative models, e.g. RBM [8], DBM [28] and VAE [14], have been proposed, these models face the difficulty of intractable functions or the difficulty of intractable inference, which in turn restricts the effectiveness of these models.

Recently, Generative adversarial networks (GANs) [6] have demonstrated impressive performance for unsupervised learning tasks. Unlike other deep generative models which usually adopt approximation methods for intractable functions or inference, GANs do not require any approximation and can be trained end-to-end through the differentiable networks. The basic idea of GANs is to simultaneously train a discriminator and a generator: the discriminator aims to distinguish between real samples and generated samples; while the generator tries to generate fake samples as real as possible, making the discriminator believe that the fake samples are from real data. So far, plenty of works have shown that GANs can play a significant role in various tasks, such as image generation [21], image super-resolution [16], and semi-supervised learning [29].

In spite of the great progress for GANs in image generation, the quality of generated images by GANs is still limited for some realistic tasks. Regular GANs adopt the sigmoid cross entropy loss function for the discriminator [6]. We argue that this loss function, however, will lead to the problem of vanishing gradients when updating the generator using the fake samples that are on the correct side of the decision boundary, but are still far from the real data. As Figure 1(b) shows, when we use the fake samples (in magenta) to update the generator by making the discriminator believe they are from real data, it will cause almost no error because they are on the correct side, i.e., the real data side, of the decision boundary. However, these samples are still far from the real data and we want to pull them close to the real data. Based on this observation, we propose the Least Squares Generative Adversarial Networks (LSGANs) which adopt the least squares loss function for the discrim-



(a) (b) (c)

Figure 1. Illustration of different behaviors of two loss functions. (a): Decision boundaries of two loss functions. Note that the decision boundary should go across the real data distribution for a successful GANs learning. Otherwise, the learning process is saturated. (b): Decision boundary of the sigmoid cross entropy loss function. The orange area is the side of real samples and the blue area is the side of fake samples. It gets very small errors for the fake samples (in magenta) when updating G as they are on the correct side of the decision boundary. (c): Decision boundary of the least squares loss function. It penalizes the fake samples (in magenta), and as a result, it forces the generator to generate samples toward decision boundary.

inator. The idea is simple yet powerful: the least squares loss function is able to move the fake samples toward the decision boundary, because the least squares loss function penalizes samples that lie in a long way on the correct side of the decision boundary. As Figure 1(c) shows, the least squares loss function will penalize the fake samples (in magenta) and pull them toward the decision boundary even though they are correctly classified. Based on this property, LSGANs are able to generate samples that are closer to real data.

Another benefit of LSGANs is the improved stability of learning process. Generally speaking, training GANs is a difficult issue in practice because of the instability of GANs learning [25]. Recently, several papers have pointed out that the instability of GANs learning is partially caused by the objective function [2, 19, 24]. Specifically, minimizing the objective function of regular GAN suffers from vanishing gradients, which makes it hard to update the generator. LSGANs can relieve this problem because LSGANs penalize samples based on their distances to the decision boundary, which generates more gradients to update the generator. Recently, Arjovsky *et al*. [2] have proposed a method to evaluate the stability of GANs learning by excluding batch normalization [11]. Following this method for evaluating the stability, we find that LSGANs are also able to converge to a relatively good state without batch normalization.

Our contributions in this paper can be summarized as follows:

- We propose LSGANs which adopt least squares loss function for the discriminator. We show that minimizing the objective function of LSGAN yields minimizing the Pearson χ^2 divergence.
- We evaluate LSGANs on LSUN and CIFAR-10 datasets and the experimental results demonstrate that LSGANs can generate more realistic images than regular GANs. Two comparison experiments for evaluating training stability are also conducted to prove the stability of LSGANs.
- We apply conditional LSGANs to the Chinese character generation. We evaluate it on a handwritten Chinese character dataset with 3740 classes. The proposed model is able to generate readable Chinese characters.

The rest of this paper is organized as follows. Section 2 briefly reviews related work of generative adversarial networks. The proposed method is introduced in Section 3, and experimental results are presented in Section 4. Finally, we conclude the paper in Section 5.

2. Related Work

Generative Adversarial Networks (GANs) were proposed by Goodfellow *et al*. [6], who explained the theory of GANs learning based on a game theoretic scenario. Showing the powerful capability for unsupervised tasks, GANs have been applied to many specific tasks, like image generation [4], image super-resolution [16], text to image synthesis [26], and image to image translation [12]. By combining the traditional content loss and the adversarial loss, super-resolution generative adversarial networks [16] achieve state-of-the-art performance for the task of image super-resolution. Reed *et al*. [26] proposed a model to synthesize images given text descriptions based on the conditional GANs [20]. Isola *et al*. [12] also used the conditional GANs to transfer images from one representation to another. In addition to unsupervised learning tasks, GANs also show the potential for semi-supervised learning tasks. Salimans *et al*. [29] proposed a GAN-based framework for semi-supervised learning, in which the discriminator not only outputs the probability that an input image is from real

data but also outputs the probabilities of belonging to each class.

Despite the great successes GANs have achieved, improving the quality of generated images is still a challenge. A lot of works have been proposed to improve the quality of images for GANs. Radford *et al*. [25] first introduced convolutional layers to GANs architecture, and proposed a network architecture called deep convolutional generative adversarial networks (DCGANs). Denton *et al*. [5] proposed another framework called Laplacian pyramid of generative adversarial networks (LAPGANs). They constructed a Laplacian pyramid to generate high-resolution images starting from low-resolution images. Further, Salimans *et al*. [29] proposed a technique called feature matching to get better convergence. The idea is to make the generated samples match the statistics of the real data by minimizing the mean square error on an intermediate layer of the discriminator.

Another critical issue for GANs is the stability of learning process. Many works have been proposed to address this problem by analyzing the objective functions of GANs [2, 3, 19, 23, 24]. Viewing the discriminator as an energy function, [33] used an auto-encoder architecture to improve the stability of GANs learning. To make the generator and the discriminator be more balanced, Metz *et al*. [19] created a unrolled objective function to enhance the generator. Che *et al*. [3] incorporated a reconstruction module and use the distance between real samples and reconstructed samples as a regularizer to get more stable gradients. Nowozin *et al*. [23] pointed out that the objective of the original GAN [6] which is related to Jensen-Shannon divergence is a special case of divergence estimation, and generalized it to arbitrary f-divergences [22]. Arjovsky *et al*. [2] extended this by analyzing the properties of four different divergences or distances over two distributions and concluded that Wasserstein distance is nicer than Jensen-Shannon divergence. Qi [24] proposed the Loss-Sensitive GAN whose loss function is based on the assumption that real samples should have smaller losses than fake samples and proved that this loss function has non-vanishing gradient almost everywhere.

3. Method

In this section, we first review the formulation of GANs briefly. Next, we present the LSGANs along with their benefits in Section 3.2. Finally, two model architectures of LSGANs are introduced in 3.3.

3.1. Generative Adversarial Networks

The learning process of the GANs is to train a discriminator D and a generator G simultaneously. The target of G is to learn the distribution p_g over data $\boldsymbol{x}$. G starts from sampling input variables $\boldsymbol{z}$ from a uniform or Gaussian distribution $p_z(\boldsymbol{z})$, then maps the input variables $\boldsymbol{z}$ to data space $G(\boldsymbol{z};\theta_g)$ through a differentiable network. On the other hand, D is a classifier $D(\boldsymbol{x};\theta_d)$ that aims to recognize whether an image is from training data or from G. The minimax objective for GANs can be formulated as follows:

$$\begin{aligned}\min_G \max_D V_{\text{GAN}}(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] \\ +\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1-D(G(\boldsymbol{z})))].\end{aligned} \tag{1}$$

3.2. Least Squares Generative Adversarial Networks

Viewing the discriminator as a classifier, regular GANs adopt the sigmoid cross entropy loss function. As stated in Section 1, when updating the generator, this loss function will cause the problem of vanishing gradients for the samples that are on the correct side of the decision boundary, but are still far from the real data. To remedy this problem, we propose the Least Squares Generative Adversarial Networks (LSGANs). Suppose we use the a-b coding scheme for the discriminator, where a and b are the labels for fake data and real data, respectively. Then the objective functions for LSGANs can be defined as follows:

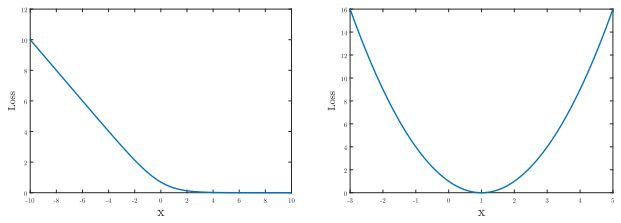
$$\begin{aligned}\min_D V_{\text{LSGAN}}(D) =& \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[(D(\boldsymbol{x})-b)^2\right] \\ &+\frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-a)^2\right] \\ \min_G V_{\text{LSGAN}}(G) =& \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-c)^2\right],\end{aligned} \tag{2}$$

where c denotes the value that G wants D to believe for fake data.

3.2.1 Benefits of LSGANs

The benefits of LSGANs can be derived from two aspects. First, unlike regular GANs which cause almost no loss for samples that lie in a long way on the correct side of the decision boundary (Figure 1(b)), LSGANs will penalize those samples even though they are correctly classified (Figure 1(c)). When we update the generator, the parameters of the discriminator are fixed, i.e., the decision boundary is fixed. As a result, the penalization will make the generator to generate samples toward the decision boundary. On the other hand, the decision boundary should go across the manifold of real data for a successful GANs learning. Otherwise, the learning process will be saturated. Thus moving the generated samples toward the decision boundary leads to making them be closer to the manifold of real data.

Second, penalizing the samples lying a long way to the decision boundary can generate more gradients when updating the generator, which in turn relieves the problem of



(a) (b)

Figure 2. (a): The sigmoid cross entropy loss function. (b): The least squares loss function.

vanishing gradients. This allows LSGANs to perform more stable during the learning process. This benefit can also be derived from another perspective: as shown in Figure 2, the least squares loss function is flat only at one point, while the sigmoid cross entropy loss function will saturate when x is relatively large.

3.2.2 Relation to Pearson χ^2 Divergence

In the original GAN paper [6], the authors has shown that minimizing Equation 1 yields minimizing the Jensen-Shannon divergence:

$$C(G) = KL\left(p_{\text{data}} \middle\| \frac{p_{\text{data}} + p_g}{2}\right) + KL\left(p_g \middle\| \frac{p_{\text{data}} + p_g}{2}\right) - \log(4). \tag{3}$$

Here we also explore the relation between LSGANs and f-divergence. Consider the following extension of Equation 2:

$$\begin{aligned} \min_D V_{\text{LSGAN}}(D) =& \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[(D(\boldsymbol{x}) - b)^2\right] \\ &+ \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z})) - a)^2\right] \\ \min_G V_{\text{LSGAN}}(G) =& \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[(D(\boldsymbol{x}) - c)^2\right] \\ &+ \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z})) - c)^2\right]. \end{aligned} \tag{4}$$

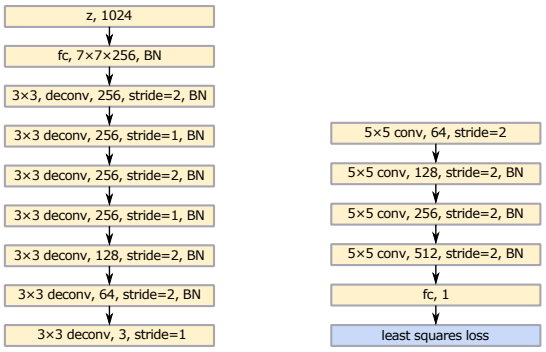
Note that adding the term $\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[(D(\boldsymbol{x}) - c)^2]$ to $V_{\text{LSGAN}}(G)$ does not change the optimal values since this term does not contain parameters of G.

We first derive the optimal discriminator D for a fixed G as below :

$$D^*(\boldsymbol{x}) = \frac{bp_{\text{data}}(\boldsymbol{x}) + ap_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}. \tag{5}$$

The proof of Equation 5 can be found in the Appendix.

In the following equations we use p_{d} to denote p_{data} for simplicity. Then we can reformulate $V_{\text{LSGAN}}(G)$ in Equation 4 as follows:



(a) (b)

Figure 3. Model architecture. "$K \times K$, conv/deconv, C, stride = S" denotes a convolutional/deconvolutional layer with $K \times K$ kernel, C output filters and stride = S. The layer with BN means that the layer is followed by a batch normalization layer. "fc, N" denotes a fully-connected layer with N output nodes. The activation layers are omitted. (a): The generator. (b): The discriminator.

$$\begin{aligned} 2C(G) &= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{d}}}\left[(D^*(\boldsymbol{x}) - c)^2\right] + \mathbb{E}_{\boldsymbol{z}\sim p_z}\left[(D^*(G(\boldsymbol{z})) - c)^2\right] \\ &= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{d}}}\left[(D^*(\boldsymbol{x}) - c)^2\right] + \mathbb{E}_{\boldsymbol{x}\sim p_g}\left[(D^*(\boldsymbol{x}) - c)^2\right] \\ &= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{d}}}\left[\left(\frac{bp_{\text{d}}(\boldsymbol{x}) + ap_g(\boldsymbol{x})}{p_{\text{d}}(\boldsymbol{x}) + p_g(\boldsymbol{x})} - c\right)^2\right] \\ &+ \mathbb{E}_{\boldsymbol{x}\sim p_g}\left[\left(\frac{bp_{\text{d}}(\boldsymbol{x}) + ap_g(\boldsymbol{x})}{p_{\text{d}}(\boldsymbol{x}) + p_g(\boldsymbol{x})} - c\right)^2\right] \\ &= \int_{\mathcal{X}} p_{\text{d}}(\boldsymbol{x})\left(\frac{(b-c)p_{\text{d}}(\boldsymbol{x}) + (a-c)p_g(\boldsymbol{x})}{p_{\text{d}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right)^2 \text{d}x \\ &+ \int_{\mathcal{X}} p_g(\boldsymbol{x})\left(\frac{(b-c)p_{\text{d}}(\boldsymbol{x}) + (a-c)p_g(\boldsymbol{x})}{p_{\text{d}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right)^2 \text{d}x \\ &= \int_{\mathcal{X}} \frac{\left((b-c)p_{\text{d}}(\boldsymbol{x}) + (a-c)p_g(\boldsymbol{x})\right)^2}{p_{\text{d}}(\boldsymbol{x}) + p_g(\boldsymbol{x})} \text{d}x \\ &= \int_{\mathcal{X}} \frac{\left((b-c)(p_{\text{d}}(\boldsymbol{x}) + p_g(\boldsymbol{x})) - (b-a)p_g(\boldsymbol{x})\right)^2}{p_{\text{d}}(\boldsymbol{x}) + p_g(\boldsymbol{x})} \text{d}x. \end{aligned} \tag{6}$$

If we set $b - c = 1$ and $b - a = 2$, then

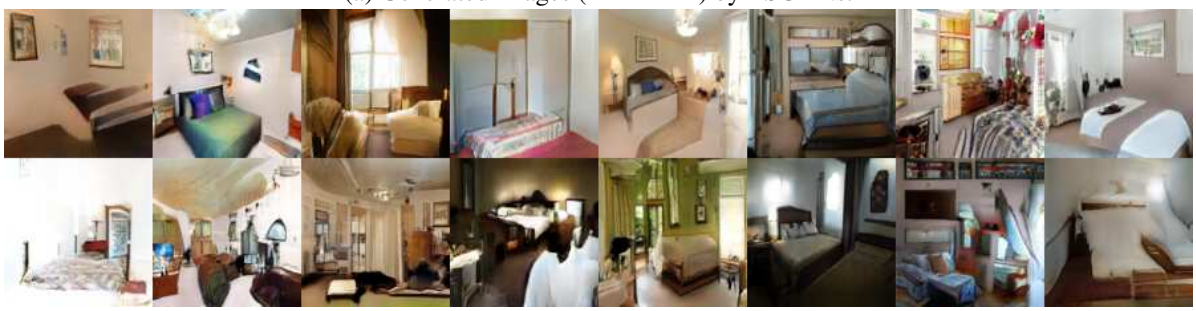
$$\begin{aligned} 2C(G) &= \int_{\mathcal{X}} \frac{\left(2p_g(\boldsymbol{x}) - (p_{\text{d}}(\boldsymbol{x}) + p_g(\boldsymbol{x}))\right)^2}{p_{\text{d}}(\boldsymbol{x}) + p_g(\boldsymbol{x})} \text{d}x \\ &= \chi^2_{\text{Pearson}}(p_{\text{d}} + p_g \| 2p_g), \end{aligned} \tag{7}$$

where χ^2_{Pearson} is the Pearson χ^2 divergence. Thus minimizing Equation 4 yields minimizing the Pearson χ^2 divergence between $p_{\text{d}} + p_g$ and $2p_g$ if a, b, and c satisfy the condtions of $b - c = 1$ and $b - a = 2$.

3.2.3 Parameters Selection

One method to determine the values of a, b, and c in Equation 2 is to satisfy the conditions of $b - c = 1$ and $b - a = 2$, such that minimizing Equation 2 yields minimizing the

(a) Generated images (112×112) by LSGANs.



(b) Generated images (112×112) by DCGANs.

(b) Generated images (64×64) by DCGANs (reported in [25]).

Figure 4. Generated images on LSUN-bedroom.

Pearson χ^2 divergence between $p_{\text{d}} + p_g$ and $2p_g$. For example, by setting $a = -1$, $b = 1$, and $c = 0$, we get the following objective functions:

$$\begin{aligned}\min_D V_{\text{LSGAN}}(D) =& \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[(D(\boldsymbol{x})-1)^2\right] \\ &+\frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))+1)^2\right] \quad (8)\\ \min_G V_{\text{LSGAN}}(G) =& \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z})))^2\right].\end{aligned}$$

Another method is to make G generate samples as real as possible by setting $c = b$. For example, by using the 0-1 binary coding scheme, we get the following objective functions:

$$\begin{aligned}\min_D V_{\text{LSGAN}}(D) =& \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[(D(\boldsymbol{x})-1)^2\right] \\ &+\frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z})))^2\right] \quad (9)\\ \min_G V_{\text{LSGAN}}(G) =& \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-1)^2\right].\end{aligned}$$

In practice, we observe that Equation 8 and Equation 9 show similar performance. Thus either one can be selected. In the following sections, we use Equation 9 to train the models.

3.3. Model Architectures

The first model we have designed is shown in Figure 3, which is motivated by the VGG model [30]. Compared with the architecture in [25], two stride=1 deconvolutional layers are added after the top two deconvolutional layers. The architecture of the discriminator is identical to the one in [25] except for the usage of the least squares loss function. Following DCGANs, ReLU activations and LeakyReLU activations are used for the generator and the discriminator, respectively.

The second model we have designed is for tasks with lots of classes, for example, Chinese characters. For Chinese characters, we find that training GANs on multiple classes is not able to generate readable characters. The reason is that there are multiple classes in the input, but only one class in the output. As stated in [9], there should be a deterministic relationship between input and output. One way to solve this problem is to use the conditional GANs [20] because conditioning on the label information creates the deterministic relationship between input and output. However, directly conditioning on the one-hot encoding label vector with thousands of classes is infeasible in terms of memory cost and computational time cost. We use a linear mapping layer to reduce the dimensionality of the label vector. For

(a) Church outdoor.

(b) Dining room.

(c) Kitchen.

(d) Conference room.

Figure 5. Generated images on different scene datasets.

the generator, the label vector is concatenated to the noise input layer. For the discriminator, the label vector is concatenated to all the convolutional layers and fully-connected layers. The layers to be concatenated are determined empirically.

4. Experiments

In this section, we first present the details of datasets and implementation. Next, we present the results of the qualitative evaluation and quantitative evaluation about LSGANs. Then we compare the stability between LSGANs and regular GANs by two comparison experiments. Finally, we evaluate LSGANs on a handwritten Chinese character dataset which contains 3740 classes.

4.1. Datasets and Implementation Details

We evaluate LSGANs on three datasets: LSUN [32], CIFAR-10 [15], and HWDB1.0 [17]. The implementation of our proposed models is based on a public implementation of DCGANs[1] using TensorFlow [1]. For LSUN, the learning rate is set to 0.001. For CIFAR-10 and HWDB1.0, the learning rate is set to 0.0002. Following DCGANs, β_1 for Adam optimizer is set to 0.5. Our implementation is available at https://github.com/xudonmao/LSGAN.

4.2. Qualitative Evaluation

We train LSGANs and DCGANs with the same network architecture (Figure 3) and same resolution (112×112) on LSUN-bedroom dataset. The generated images by the two methods are presented in Figure 4. Compared with the images generated by DCGANs, the texture detail (e.g., the textures of beds) of the images generated by LSGANs is more exquisite and the images generated by LSGANs looks sharper.

We also train LSGANs on four other scene datasets including church, dining room, kitchen, and conference room. The results of LSGANs trained on these four scene datasets are shown in Figure 5.

[1]https://github.com/carpedm20/DCGAN-tensorflow

(a) LSGANs: without BN in G using Adam.

(b) Regular GANs: without BN in G using Adam.

(c) LSGANs: without BN in G and D using RMSProp.

(d) Regular GANs: without BN in G and D using RMSProp.

Figure 6. Comparison experiments by excluding batch normalization (BN).

4.3. Quantitative Evaluation

4.3.1 Inception Score on CIFAR-10

We train LSGANs and DCGANs with the same network architecture on CIFAR-10 and use the models to randomly generate $50,000$ images for calculating the inception scores [29]. The evaluated inception scores of LSGANs and DCGANs are shown in Table 1. As we observe that the inception scores vary for different trained models, the reported inception scores in Table 1 are averaged over 10 different trained models for both LSGANs and DCGANs. For this quantitative evaluation of inception score, LSGANs show comparable performance to DCGANs.

Method	Inception Score
DCGAN (reported in [10])	6.16
DCGAN	6.22
LSGAN (ours)	**6.47**

Table 1. Inception scores on CIFAR-10.

4.3.2 Human Subjective Study

To further evaluate the performance of LSGANs, we conduct a human subjective study using the generated bedroom images (112×112) from LSGANs and DCGANs with the same network architectures. We randomly construct image pairs, where one image is from LSGANs and the other one is from DCGANs. We ask Amazon Mechanical Turk annotators to judge which image looks more realistic. With 4,000 votes totally, DCGANs get 43.6% votes and LSGANs get 56.4% votes. LSGANs get 12.8% more votes than DCGANs.

4.4. Stability Comparison

As stated in Section 3.2.1, one benefit of LSGANs is the improved stability. Here we present two comparison experiments to compare the stability between LSGANs and regular GANs.

One is to follow the comparison method in [2]. Based on the network architecture presented in [25], two architectures are designed to compare the stability. The first one is to exclude the batch normalization for the generator (BN_G for short), and the second one is to exclude the batch normalization for both the generator and discriminator (BN_{GD} for short). As pointed out in [2], the selection of optimizer is critical to the model performance. Thus we evaluate the two architectures with two optimizers, Adam [13] and RMSProp [31]. In summary, we have four training settings: (1) BN_G with Adam, (2) BN_G with RMSProp, (3) BN_{GD} with Adam, and (4) BN_{GD} with RMSProp.

We train the above models on the LSUN-bedroom dataset using regular GANs and LSGANs separately and have the following four major observations. First, for BN_G with Adam, there is a chance for LSGANs to generate relatively good quality images. We test 10 times, and 5 of those succeeds to generate relatively good quality images. But for regular GANs, we never observe successful learning. Regular GANs suffer from a severe degree of mode collapse. The generated images by LSGANs and regular GANs are shown in Figure 6. Second, for BN_{GD} with RMSProp, as Figure 6 shows, LSGANs generate higher quality images than regular GANs which have a slight degree of mode collapse. Third, LSGANs and regular GANs have similar performances for BN_G with RMSProp and BN_{GD} with Adam. Specifically, for BN_G with RMSProp, both LSGANs and regular GANs are able to generate relatively good images. For BN_{GD} with Adam, both have a slight degree of mode collapse. Last, RMSProp performs more stable than Adam since regular GANs learn to generate relatively good images for BN_G with RMSProp, but fail to learn with Adam.

Another experiment is to evaluate on a Gaussian mixture distribution dataset, which is designed in literature [19]. We train LSGANs and regular GANs on a 2D mixture of

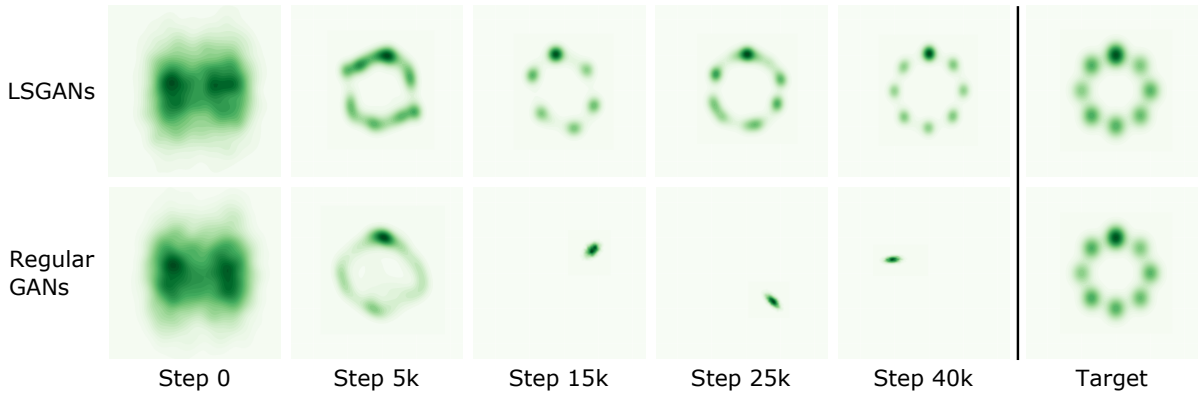


Figure 7. Dynamic results of Gaussian kernel estimation for LSGANs and regular GANs. The final column shows the real data distribution.

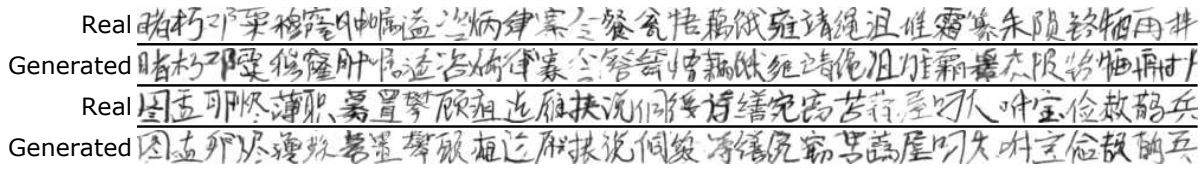


Figure 8. Generated images of handwritten Chinese characters by LSGANs. For row 1 and row 2, the images in the same column belong to the same class of characters. Row 3 and row 4 are also with this condition. The generated characters are readable.

8 Gaussian dataset using a simple network architecture, where both the generator and the discriminator contain three fully-connected layers. Figure 7 shows the dynamic results of Gaussian kernel density estimation. We can see that regular GANs suffer from mode collapse starting at step 15k. They generate samples around a single valid mode of the data distribution. But LSGANs learn the Gaussian mixture distribution successfully.

4.4.1 Suggestions in Practice

During the learning processes of LSGANs for tasks which are difficult to train, we observe that LSGANs learn to generate good quality images successfully at the first several training epochs, but sometimes suffer from mode collapse at last. Although LSGANs may suffer from mode collapse at last, we can still select a good model in the middle of the training process. We also observe that the quality of generated images by LSGANs may shift between good and bad during the training process. Based on the above two observations, we suggest to keep a record of generated images at every thousand or hundred iterations and select the model manually by checking the image quality.

4.5. Handwritten Chinese Characters

We also train a conditional LSGAN model (described in Section 3.3) on a handwritten Chinese character dataset which contains 3740 classes. LSGANs learn to generate readable Chinese characters successfully, and some randomly selected characters are shown in Figure 8. We have two major observations from Figure 8. First, the generated characters by LSGANs are readable. Second, we can get the correct labels of the generated images through label vectors, which can be used for further applications such as data augmentation.

5. Conclusions and Future Work

In this paper, we have proposed the Least Squares Generative Adversarial Networks (LSGANs). The experimental results show that LSGANs generate higher quality images than regular GANs. Two comparison experiments for evaluating the stability are also conducted and the results demonstrate that LSGANs perform more stable than regular GANs. Furthermore, we propose a conditional LSGAN model for Chinese character generation, which is evaluated on a handwritten Chinese character dataset with 3740 classes. Based on the present findings, we hope to extend LSGANs to more complex datasets such as ImageNet in the future. Instead of pulling the generated samples toward the decision boundary, designing a method to pull the generated samples toward the real data directly is also worth our further investigation.

Acknowledgments

This work is supported by a research grant (project number: 9360153) and a special grant (account number: 9610367) from City University of Hong Kong.

References

[1] M. Abadi, A. Agarwal, P. Barham, and et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

[2] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv:1701.07875*, 2017.

[3] T. Che, Y. Li, A. P. Jacob, Y. Bengio, and W. Li. Mode regularized generative adversarial networks. *arXiv:1612.02136*, 2016.

[4] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2172–2180, 2016.

[5] E. Denton, S. Chintala, A. Szlam, and R. Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1486–1494, 2015.

[6] 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 (NIPS)*, pages 2672–2680, 2014.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[8] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504 – 507, 2006.

[9] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, July 1989.

[10] X. Huang, Y. Li, O. Poursaeed, J. Hopcroft, and S. Belongie. Stacked generative adversarial networks. *arXiv:1612.04357*, 2016.

[11] S. Ioffe and C. Szegedy. Generative adversarial text-to-image synthesis. In *Proceedings of The 33rd International Conference on Machine Learning (ICML)*, 2015.

[12] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *arXiv:1611.07004*, 2016.

[13] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[14] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

[15] A. Krizhevsky. Learning multiple layers of features from tiny images. *Tech Report*, 2009.

[16] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. *arXiv:1609.04802*, 2016.

[17] C.-L. Liu, F. Yin, Q.-F. Wang, and D.-H. Wang. Icdar 2011 chinese handwriting recognition competition. In *Proceedings of the 2011 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1464–1469, 2011.

[18] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[19] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein. Unrolled generative adversarial networks. *arXiv:1611.02163*, 2016.

[20] M. Mirza and S. Osindero. Conditional Generative Adversarial Nets. *arXiv:1411.1784*, 2014.

[21] A. Nguyen, J. Yosinski, Y. Bengio, A. Dosovitskiy, and J. Clune. Plug & play generative networks: Conditional iterative generation of images in latent space. *arXiv:1612.00005*, 2016.

[22] X. Nguyen, M. J. Wainwright, and M. I. Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861, 2010.

[23] S. Nowozin, B. Cseke, and R. Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. *arXiv:1606.00709*, 2016.

[24] G.-J. Qi. Loss-sensitive generative adversarial networks on lipschitz densities. *arXiv:1701.06264*, 2017.

[25] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2015.

[26] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text-to-image synthesis. In *Proceedings of The 33rd International Conference on Machine Learning (ICML)*, 2016.

[27] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems 28*, pages 91–99, 2015.

[28] R. Salakhutdinov and G. Hinton. Deep Boltzmann machines. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 5, pages 448–455, 2009.

[29] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, and X. Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2226–2234, 2016.

[30] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

[31] T. Tieleman and G. Hinton. Lecture 6.5—RMSProp: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 2012.

[32] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv:1506.03365*, 2015.

[33] J. Zhao, M. Mathieu, and Y. LeCun. Energy-based Generative Adversarial Network. *arXiv:1609.03126*, 2016.