Least Squares Generative Adversarial Networks

Xudong Mao*[1], Qing Li†[1], Haoran Xie‡[2], Raymond Y.K. Lau§[3], and Z Wang¶[4]

[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 (OPTIMAL), Northwestern Polytechnical University, China

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. This loss function, however, may lead to the vanishing gradient problem during the learning process. To overcome such problem, here we propose the Least Squares Generative Adversarial Networks (LSGANs) that 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 performs more stable during the learning process. We evaluate the LSGANs on five scene datasets and the experimental results demonstrate that the generated images by LSGANs look more realistic 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 [1], object detection [2]

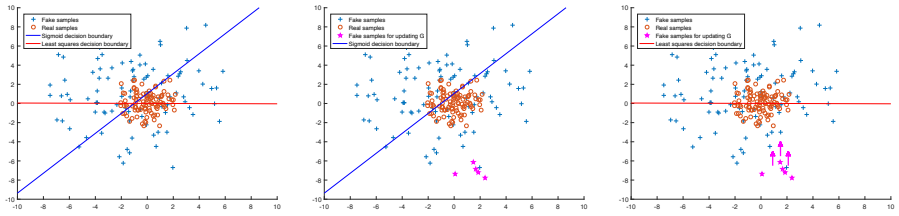
*xudonmao@gmail.com
†itqli@cityu.edu.hk
‡hrxie2@gmail.com
§raylau@cityu.edu.hk
¶zhenwang0@gmail.com

and segmentation [3]. 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 [4], DBM [5] and VAE [6], have been proposed, these models have the difficulty of intractable functions or the difficulty of intractable inference, which in turn restricts the effectiveness of these models.

Generative adversarial networks (GANs) [7] have demonstrated their effectiveness 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 training data. So far, plenty of works have proven that GANs plays a significant role in various tasks, such as image generation [8], image super-resolution [9], and semi-supervised learning [10].

In spite of 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 [7]. We argue that this loss function 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 motivation, we propose the Least Squares Generative Adversarial Networks (LSGANs) which adopt the least squares loss function for the discriminator. 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, training GANs is a difficult issue in practice because of the instability of GANs learning [11, 12]. To overcome this point, various well-designed networks have been suggested, for example, DCGANs [13]. One key point that [13] suggested is to use batch normalization [14] for both the discriminator and the generator. Recently, several papers have pointed out that the instability of GANs learning is partially caused by the objective function [11, 12]. Arjovsky



(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. It gets very small errors for the fake samples (in magenta) for updateing G as they are on the correct side of the decision boundary. (c): Decision boundary of the least squares loss function. It penalize the fake samples (in magenta), and as a result, it forces the generator to generate samples toward decision boundary.

et al. [11] proposed a Wasserstein-based objective function and illustrated the stability of their model by excluding batch normalization for the generator. By following this method for comparing the stability of learning process, we find that LSGANs are also able to converge to a relatively good state without batch normalization in both the discriminator and the generator. Furthermore, training LSGANs is faster to converge than Wasserstein GANs as Wasserstein GANs require multiple updates for the discriminator.

Our contributions can be summarized as follows:

- We propose Least Squares Generative Adversarial Networks (LSGANs) that adopt least squares loss function for the discriminator. We also show that minimizing the objective function of LSGAN yields minimizing the Pearson χ^2 divergence. The experimental results demonstrate that LSGANs can generate more realistic images than regular GANs. Numerous comparison experiments are also conducted to prove the stability of LSGANs.

- Two network architectures of LSGANs are designed. The first one is for image generation with 112×112 resolution, which is evaluated on various kinds of scene datasets. The experimental results show that LSGANs generate higher quality images than the current state-of-the-art method. The second one is for tasks with a lot of classes. We evaluate it on a handwritten Chinese characters dataset with 3470 classes and the proposed model is able to generate elegant images.

The rest of this paper is organized as follows. Section 2 briefly reviews related work of generative adversarial networks. The proposed method is introduced

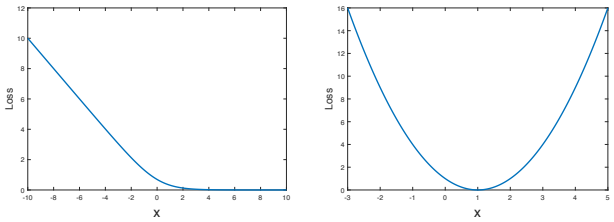
in Section 3, and experimental results are presented in Section 4. Finally, we conclude the paper in Section 5.

2 Related Work

Deep generative models attempt to capture the probability distributions over the given data. Restricted Boltzmann Machines (RBMs), one type of deep generative models, are the basis of many other hierarchical models, and they have been used to model the distributions of images [15] and documents [16]. Deep Belief Networks (DBNs) [17] and Deep Boltzmann Machines (DBMs) [5] are extended from the RBMs. The most successful application of DBNs is for image classification [17], where DBNs are used to extract feature representations. However, RBMs, DBNs and DBMs all have the difficulties of intractable partition functions or intractable posterior distributions, which thus use the approximation methods to learn the models. Another important deep generative model is Variational Autoencoders (VAE) [6], a directed model, which can be trained with gradient-based optimization methods. But VAEs are trained by maximizing the variational lower bound, which may lead to the blurry problem of generated images.

Recently, Generative Adversarial Networks (GANs) have been proposed by Goodfellow *et al.* [7], who explained the theory of GANs learning based on a game theoretic scenario. Compared with the above models, training GANs does not require any approximation method. Like VAEs, GANs also can be trained through differentiable networks. Showing the powerful capability for unsupervised tasks, GANs have been applied to many specific tasks, like image generation [18], image super-resolution [9], text to image synthesis [19] and image to image translation [20]. By combining the traditional content loss and the adversarial loss, super-resolution generative adversarial networks [9] achieve state-of-the-art performance for the task of image super-resolution. Reed *et al.* [19] proposed a model to synthesize images given text descriptions based on the conditional GANs [21]. Isola *et al.* [20] also use the conditional GANs to transfer images from one representation to another. In addition to unsupervised learning tasks, GANs also show potential for semi-supervised learning tasks. Salimans *et al.* [10] 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.* [13] first introduced convolutional layers to GANs architecture, and proposed a network architecture called deep convolutional generative adversarial networks (DCGANs). Denton *et al.* [22] proposed another framework called Laplacian pyramid of generative adversarial networks (LAPGANs). They construct a Laplacian pyramid to generate high-resolution images starting from low-resolution images. Further, Salimans *et al.* [10] proposed a technique called feature matching to get better



(a) (b)

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

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 [11, 23, 12, 24, 25]. Viewing the discriminator as an energy function, [26] uses an auto-encoder architecture to improve the stability of GANs learning. To make the generator and the discriminator be more balanced, Metz *et al.* [12] create a unrolled objective function to enhance the generator. Che *et al.* [23] incorporate a reconstruction module and use the distance between real samples and reconstructed samples as a regularizer to get more stable gradients. Nowozin *et al.* [24] point out that the objective of original GAN [7] which is related to Jensen-Shannon divergence is a special case of divergence estimation, and generalize it to arbitrary f-divergences [27]. [11] extends this by analyzing the properties of four different divergences or distances over two distributions and concludes that Wasserstein distance is nicer than Jensen-Shannon divergence.

3 Method

In this section, we first review the formulation of GANs briefly. Then 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:

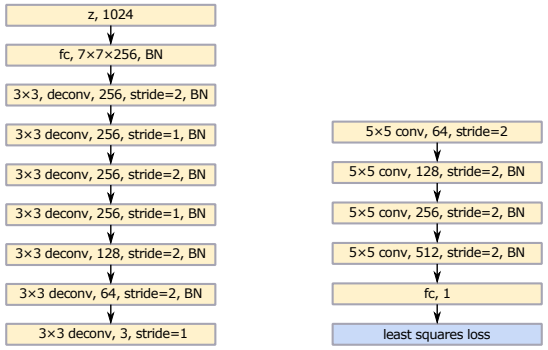


Figure 3: Model architecture. "$K \times K$, conv/deconv, C, stride $= S$" denotes a convolutional/deconvolutional layer with $K \times K$ kernel, C output filters and stride $= S$. The layer with BN means that the layer is followed by a batch normalization layer. "fc, N" denotes a fully-connected layer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

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

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, this loss function will cause the problem of vanishing gradients for the samples that lie in a long way on the correct side of the decision boundary when updating the generator. 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 objectives for LSGANs can be defined as follows:

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

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

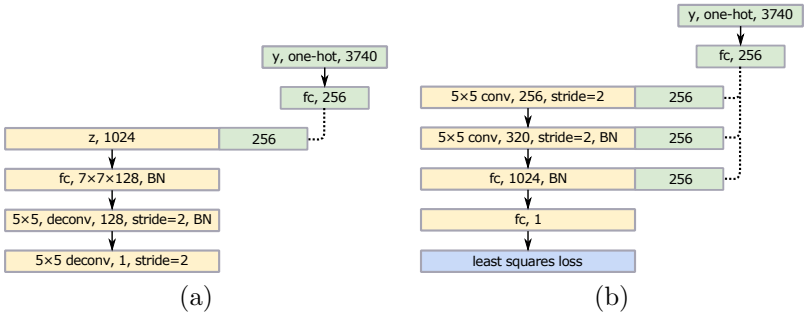


Figure 4: Model architecture for datasets with amount of classes. (a): The generator. (b): The discriminator.

3.2.1 Benefits of LSGANs

The benefits of LSGANs can be derived from two perspectives. First, as Figure 1(b) shows, for the sigmoid cross entropy loss function, when updating the generator, there is almost no loss caused by the samples that lie in a long way on the correct side of the decision boundary. However, these samples are still far from the real data, and it is not reasonable to cause no loss for these samples. Unlike regular GANs, the least squares loss function will penalize the samples that lie in a long way of the decision boundary even though they are correctly classified as Figure 2(b) shows. 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 real data distribution for a successful GANs learning. Otherwise, the learning process will be saturated. Thus moving the generated samples toward the decision boundary lead to making them be closer to the real data. In summary, the least squares loss function can help LSGANs to generate samples that are closer to the real data. Second, as shown in Figure 2, the least squares loss function is flat only at one point. However, the sigmoid cross entropy loss function will saturate when x is relatively large. This property makes LSGANs more stable than regular GANs for the learning process.

3.2.2 Reltation to f-divergence

In the original GAN paper [7] 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:

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

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 as this term does not contain parameters of G.

First the optimal discriminator D can be derived as follows for fixed G:

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

In the following equation we use p_{d} to denote p_{data} for simplicity. Then we can reformulate the Equation 4:

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

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

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

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, c satisfy the condtions $b-c=1$ 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 $b-c=1$ and $b-a=2$ such that minimizing Equation 2 yields minimizing the 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 objective functions:

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

The intuition of 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 objective functions:

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

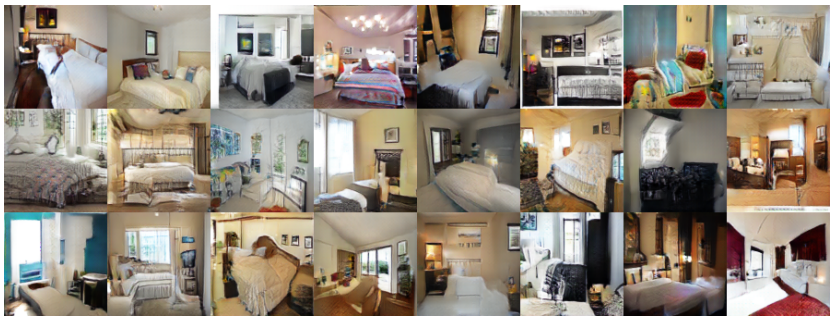
In practice, we observe that Equation 8 and Equation 9 show similar performance. Thus either one can be adopted. In the following section, we use Equation 9 to train the models.

3.3 Model Architectures

Based on the above analysis, we designed the first model which is shown in Figure 3. The design of the generator is motivated by the VGG model [28]. Compared with the architecture of DCGANs, two stride=1 deconvolutional layers are added after the top two deconvolutional layers in DCGANs. The architecture of the discriminator is identical to DCGANs except for the usage of the least squares loss function. Following DCGANs, ReLU activations are used for the generator and LeakyReLU activations are used for the discriminator.

The second model we designed is for tasks with lots of classes, for example, the Chinese characters. For Chinese characters, we find that training GANs on multiple classes is not able to generate recognizable characters. The reason is that there are multiple classes in the input but only one class in the output. As stated in [29], there should be a deterministic relationship between input and output. We propose to use the conditional GANs [21] for datasets with multiple classes because conditioning on the label information creates the deterministic relationship between input and output. However, with one-hot encoding for label information, directly conditioning on the label vectors with thousands of classes is infeasible in memory cost and computational time cost. Thus we use a linear mapping layer to map the large label vectors into small vectors. In summary, the model architecture for tasks with lots of classes is shown in Figure 4 and the corresponding objectives can be defined as follows:

$$\begin{aligned}\min_D V_{\text{LSGAN}}(D) =& \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\big[(D(\boldsymbol{x}|\Phi(\boldsymbol{y}))-1)^2\big] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\big[(D(G(\boldsymbol{z})|\Phi(\boldsymbol{y})))^2\big]. \\ \min_G V_{\text{LSGAN}}(G) =& \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\big[(D(G(\boldsymbol{z})|\Phi(\boldsymbol{y}))-1)^2\big].\end{aligned} \tag{10}$$



(a) Generated by LSGANs.

(b) Generated by DCGANs (Reported in [13]).

Figure 5: Generated images on LSUN-bedroom.

where $\Phi(\cdot)$ denotes the linear mapping function and $\boldsymbol{y}$ denotes the label vectors.

4 Experiments

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

Table 1: Statistics of the datasets.

Dataset	#Samples	#Categories
LSUN Bedroom	3,033,042	1
LSUN Church	126,227	1
LSUN Dining	657,571	1
LSUN Kitchen	2,212,277	1
LSUN Conference	229,069	1
HWDB1.0	1,246,991	3,740

(a) Church outdoor. (b) Dining room.

(c) Kitchen. (d) Conference room.

Figure 6: Generated images on different scene datasets.

4.1 Datasets and Implementation Details

We evaluate LSGANs on two datasets, LSUN [30] and HWDB1.0 [31]. The details of the two datasets are presented in Table 1. The implementation of our proposed models is based on a public implementation of DCGANs[1] using TensorFlow [32]. The learning rates for scenes and Chinese characters are set to 0.001 and 0.0002 respectively. Following DCGANs, β_1 for Adam optimizer is set to 0.5. All the codes of our implementation will be public available soon.

4.2 Scenes

We train LSGANs (Figure 3) on five scene datasets in LSUN including bedroom, kitchen, church, dining room and conference room. The generated images on LSUN-bedroom compared with DCGANs are presented in Figure 5. We have two major observations from Figure 5. First, the images generated by LSGANs looks more realistic than the ones generated by DCGANs. Second, the images generated by LSGANs are less blurry than the ones generated by DCGANs. The results trained on other scenes are shown in Figure 6.

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

(a) LSGANs. (b) Regular GANs.

(c) LSGANs. (d) Regular GANs.

Figure 7: Comparison experiments by excluding batch normalization (BN). (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.

Table 2: Whether the models suffer from model collapse?

Optimizer	BN_G Adam	BN_G RMSProp	BN_{GD} Adam	BN_{GD} RMSProp
Regular GANs	YES	NO	YES	YES
LSGANs	**NO**	NO	YES	**NO**

4.3 Stability Comparison

As stated in Section 3.2, one benefit of LSGANs is the improved stability. Here we present two kinds of comparison experiments to compare the stability between LSGANs and regular GANs.

One is to follow the comparison method in [11]. Based on the network architectures of DCGANs, 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 the to exclude the batch normalization for both the generator and the discriminator (BN_{GD} for short). We find that the selection of optimizer is critical to the performance of some specific architectures. For example, for BN_{GD}, LSGANs with RMSProp [33] can converge to a relatively good state while leading to mode collapse with Adam [34]. We list all the results for different optimizers in Table 2. We also show some images generated by LSGANs and regular GANs in Figure 7. We have three observations from Table 2 and Figure 7. First, LSGANs are able to generate relatively good images for both BN_G and BN_{GD}. Second, LSGANs perform more stable than regular GANs. Last, RMSProp performs better for both BN_G and BN_{GD} as RMSProp is more stable for very nonstationary problems.

Another one is to evaluate with a specially designed architecture as Figure 8 shows. We evaluate it on MNIST. To evaluate the capability of LSGANs, we modify the architecture in two aspects to make the discriminator easier to be saturated: (1) make the generator and the discriminator condition on label

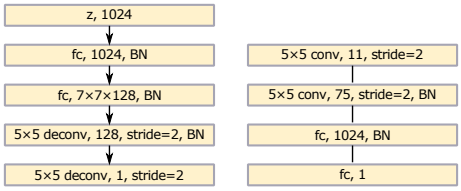


Figure 8: Special designed architecture for comparing the stability. (a) The generator. (b) The discriminator.

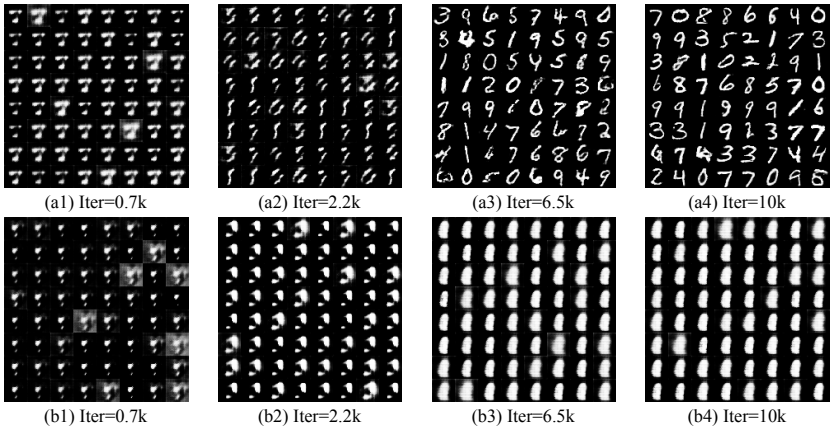


Figure 9: Generated images on MNIST. Upper: Generated by LSGANs. Lower: Generated by regular GANs.

vectors. (2) add sigmoid activation to the last layer of the discriminator. Then we train LSGANs and regular GANs on MNIST. Results are shown in Figure 9. LSGANs learn the data distribution successfully, while regular GANs suffer from model collapse.

4.4 Handwritten Chinese Characters

We also train a LSGAN model (Figure 4) on a handwritten Chinese characters dataset which contains 3740 classes. We randomly sample 64 characters from different classes as Figure 10 shows. We have two major conclusions from Figure 10. First, the generated characters by LSGANs are easy to be recognized. Second, the generated images are with correct labels. The generated images with labels can be used for further application, for example, data augmentation.

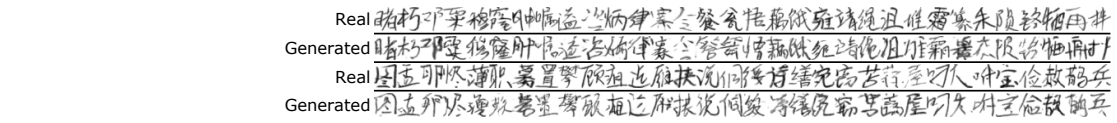


Figure 10: Generated images of handwritten Chinese characters by LSGANs.

5 Conclusions and Future Work

To conclude, here we have proposed the Least Squares Generative Adversarial Networks. Two model architectures are designed. The first one is evaluated on several scene datasets. The experimental results show that LSGANs generate higher quality images than regular GANs. The second one is evaluated on a handwritten Chinese characters dataset with 3740 classes. Besides, numerous comparison experiments for evaluating the stability are conducted and the results demonstrate that LSGANs perform more stable than regular GANs in the learning process. Based on the present findings, we hope to extend LSGANs to more complex datasets such as ImageNet in 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 attention.

References

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

[2] 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*, pp. 91–99, 2015.

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

[4] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504 – 507, 2006.

[5] R. Salakhutdinov and G. Hinton, "Deep Boltzmann machines," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, vol. 5, pp. 448–455, 2009.

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

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Ad-*

vances in Neural Information Processing Systems (NIPS), pp. 2672–2680, 2014.

[8] 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.

[9] 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.

[10] 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)*, pp. 2226–2234, 2016.

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

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

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

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

[15] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler, "Convolutional learning of spatio-temporal features," in *European Conference on Computer Vision (ECCV)*, pp. 140–153, 2010.

[16] G. E. Hinton and R. R. Salakhutdinov, "Replicated softmax: an undirected topic model," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 1607–1614, 2009.

[17] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, pp. 1527–1554, July 2006.

[18] 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)*, pp. 2172–2180, 2016.

[19] 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.

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

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

[22] 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)*, pp. 1486–1494, 2015.

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

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

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

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

[27] 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*, vol. 56, no. 11, pp. 5847–5861, 2010.

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

[29] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, July 1989.

[30] 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.

[31] 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)*, pp. 1464–1469, 2011.

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

[33] 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.

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