QUANTITATIVELY EVALUATING GANS WITH DIVERGENCES PROPOSED FOR TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative adversarial networks (GANs) have made enormous progress in terms of theory and application in the machine learning and computer vision communities. Even though substantial progress has been made, a lack of quantitative model assessment is still a serious issue. This has led to a huge number of GAN variants being proposed, with relatively little understanding of their relative abilities. In this paper, we evaluate the performance of various types of GANs using divergence and distance functions typically used only for training. We observe consistency across the various proposed metrics, and interestingly, the test-time metrics do not favour networks that use the same training-time criterion.

1 INTRODUCTION

Generative adversarial networks (GANs) aim to approximate a data distribution P, using a parameterized model distribution Q. They achieve this by jointly optimizing generative and discriminative networks (Goodfellow et al., 2014). GANs are end-to-end differentiable, and can produce exact samples from the distribution of the generative network Q. The samples of a generative network are propagated forward to a discriminative network, and the error signals are then propagated backwards to the generative network, passing through the discriminative network. This is often viewed as learning an adaptive loss function for a generative network by training a discriminative network.

The success of learning a probability distribution depends heavily on the metric used to measure the distance between P and Q. Fortunately, GAN frameworks can be formulated as a divergence or distance function of various forms (involving supremum/infimum over the function class),

$$\sup_{f \in \mathcal{F}} \mathbb{E}_{P(x)} \left[\mu\left(f(x)\right)\right] - \mathbb{E}_{Q(x)} \left[\upsilon\left(f(x)\right)\right] \tag{1}$$

for some real-valued bounded class of functions $\mathcal{F}$. Several GAN variants, such as f-GAN (Nowozin et al., 2016), Wasserstein GAN (Arjovsky et al., 2017), and Least-Square GAN (Mao et al., 2017) have the above form with specific μ and υ functions (see Table 1).

The problem is relaxed by converting the supremum operator over the function class to have the maximum operator over a parametric model class, and minimizing the generative network. Finally, the GAN is formulated as a min-max problem:

$$\min_{\theta} \max_{\phi} \mathbb{E}_{P(x)} \left[\mu\left(D_\phi(x)\right)\right] - \mathbb{E}_{Q_\theta(x)} \left[\upsilon\left(D_\phi(x)\right)\right] \tag{2}$$

where θ and ϕ are the parameters of the generative and discriminative networks, D_ϕ corresponds to the discriminative network, and $Q_\theta(x)$ corresponds to the generative network's distribution.

Many practitioners who deploy GANs are primarily concerned with sample quality, as GANs are known to generate realistic-looking sharp samples compared to other popular generative models, such as variational autoencoders (Kingma & Welling, 2014). Nonetheless, there is no principled way of finding an architecture for high quality samples. Plenty of previous probabilistic generative models were evaluated based on the point-wise likelihood of the test data, and they were trained to maximize the point-wise likelihood. However, GANs do not use a point-wise criterion during training, so there is no easy way to evaluate the test likelihood without simplifying assumptions. But more importantly, it has been shown that the likelihood is generally uninformative about the quality of samples and vice versa (Theis et al., 2015), which is what GAN practitioners care about.

Table 1: Defined μ and v functions for some GAN distance metrics. The $\mathcal{F}$ is class of real-value bounded functions and $\|\cdot\|_1$ stands for 1-Lipschitz functions. $\leq$ stands for lower bound. The a and b values are constrained to be $b - c = 1$ and $b - a = 2$, and b and a are replaced by c when minimizing generator (Mao et al., 2017).

Metric	μ	v	Function Class
$\leq$ Jensen-Shannon	$\log f$	$\log(1-f)$	$\mathcal{F}$
(Constrained) Pearson	$-(f-b)^2$	$(f-a)^2$	$\mathcal{F}$
Wasserstein	f	$-f$	$f \in \mathcal{F} : \|f\|_L \leq 1$

Evaluating the quality of samples with human vision can be time-consuming and difficult to reproduce. In order to quantitatively measure the sample quality, we propose to evaluate GANs by exploring the divergence and distance metrics that were used to train the GANs. We show that rankings produced using our proposed metrics are consistent and robust across metrics.

We use these metrics to quantitatively analyze three different families of GANs. This analysis includes measuring overall performance, as well as carrying out a sensitivity analysis with respect to various factors, such as the architecture size, noise dimension, update ratio between discriminator and generator, and number of data points. Our empirical results show that: i) the larger the GAN architecture the better the results, ii) having a generator network larger than the discriminator network does not yield good results, iii) the best ratio between discriminator and generator updates depended on the data set, and iv) the W-GAN and LS-GAN performance increases much faster than DCGAN as the number of training examples grows. Ultimately, we can tune the hyper-parameters and architectures of GANs based on our proposed method.

2 Previous Evaluation Metrics

The **Generative Adversarial Metric (Im et al., 2016a)** measures the relative performance of two GANs by measuring the likelihood ratio of the two models. Consider two GANs with their respective trained partners, $M_1 = (D_1, G_1)$ and $M_2 = (D_2, G_2)$, where G_1 and G_2 are the generators, and D_1 and D_2 are the discriminators. The hypothesis $\mathcal{H}_1$ is that M_1 is better than M_2 if G_1 fools D_2 more than G_2 fools D_1, and vice versa for the hypothesis $\mathcal{H}_0$. The likelihood-ratio is defined as:

$$\frac{p(x|y=1; M_1')}{p(x|y=1; M_2')} = \frac{p(y=1|x; D_1)p(x; G_2)}{p(y=1|x; D_2)p(x; G_1)} \tag{3}$$

where M_1' and M_2' are the swapped pairs (D_1, G_2) and (D_2, G_1), and $p(x|y=1, M)$ is the likelihood of x generated from the data distribution $p(x)$ by model M. Hence, the procedure continues by simply swapping the partners of the two GANs, where the discriminator and generator from the first GAN is partnered with the discriminator and generator from the second GAN. Then, we can measure which generator fools the opponent's discriminator more, $\frac{D_1(S_2)}{D_2(S_1)}$ where $S_1 \sim G_1$ and $S_2 \sim G_2$.

There are two main caveats to the Generative Adversarial Metric. First, the measurement is a relative comparison between the two models. Second, the metric has a constraint where the two discriminators must have an approximately similar performance on a calibration dataset, which can be difficult to satisfy in practice.

The **Inception Score (Salimans et al., 2016)** measures the performance of a model using a well-trained third party neural network. The Inception Score computes the expectation of divergence between the probability of class prediction, given samples and a prior distribution over the class,

$$\exp\left(\mathbb{E}_x KL(p(y|x) \| p(y))\right) \tag{4}$$

where the probability of class prediction given the samples is computed using a third party neural network that is trained by supervised learning. In the original paper that proposed the Inception Score, Google's inception network (Szegedy et al., 2015) was used to compute $p(y|x)$.

Inception Score is the most widely used metric to measure GAN performance. Nonetheless, it is an indirect way of measuring the generative model quality, as it considers the distance between the distribution over the class, rather than the distance between the data distribution P and the model

distribution Q. As well, it requires another neural network that is trained separately via supervised learning. We demonstrate an example of a failure case of Inception Score in the experiments section.

Log-likelihood via AIS (Wu et al., 2016) measures the log-likelihood of decoder-based models using Annealed Importance Sampling (AIS). AIS proceeds by considering many intermediate distributions that are defined by taking a weighted geometric mean between the prior (initial) distribution, $p(z)$, and the joint (target) distribution, $p_\sigma(x, z) = p(x|z)p(z)$.

The drawback of using the estimator as was done in (Wu et al., 2016) is that it makes a Gaussian observation model assumption with fixed variance. Unfortunately, neither a GANs' objectives nor architectures have any particular observation model built in, whereas variational auto-encoder training objectives have a Gaussian observable model included. In Section 4, we demonstrate that the estimate can vary considerably, sometimes depending on the size of the fixed variance. More importantly, recent work has shown that the log-likelihood does not reflect sample quality (Theis et al., 2015).

3 Evaluation Metrics

Given a generator with fixed parameters θ, we can simply compute the distance/divergence of P and Q_θ (See Algorithm 1). We can consider metrics from two widely studied distance and divergence measures, $f-$divergence (Nguyen et al., 2008) and the Integral Probability Metric (IPM) (Muller, 1997). Note that there is no overlap of the distance functions between the two classes, except for total variation distance (Sriperumbudur et al., 2009). In our experiments, we consider the following four metrics that are commonly used to train GANs:

Metric 1. Jensen-Shannon Divergence

Training a standard GAN corresponds to minimizing the lower bound of Jensen-Shannon Divergence (Goodfellow et al., 2014):

$$\max_{\varphi} \mathbb{E}_{x\sim p(x)}[\log(D_\varphi(x))] + \mathbb{E}_{z\sim p(z)}[\log(1 - D_\varphi(G_\theta(z)))] \tag{5}$$

where $p(z)$ is the prior distribution of the generative network, and $G_\theta(z)$ is a differentiable function from z to the data space represented by a neural network with parameter θ.

Metric 2. Constrained Pearson $\mathcal{X}^2$

A Least-Squares GAN corresponds to training under Pearson $\mathcal{X}^2$ divergence with constrained variables $b - a = 2$ and $b - c = 1$ (Mao et al., 2017):

$$\max_{\varphi} -\mathbb{E}_{x\sim p(x)}[(D_\varphi(x) - b)^2] - \mathbb{E}_{z\sim p(z)}[(D_\varphi(G_\theta(z) - a))^2] \tag{6}$$

where φ and θ are parameters of the discriminative and generative network, respectively. Typically a, b, c are set to $0, 1, 1$, respectively.

Metric 3. Maximum Mean Discrepancy

$$D^k_{\mathcal{H}}(P, Q) = \frac{1}{n(n-1)}\mathbb{E}_{x,x'\sim P}\left[k(x, x')\right] + \frac{1}{m(m-1)}\mathbb{E}_{z,z'\sim p(Z)}\left[k(G_\theta(z), G_\theta(z'))\right] - \frac{2}{nm}\mathbb{E}_{x\sim P, z\sim p(Z)}\left[k(x, G_\theta(z))\right] \tag{7}$$

where $\mathcal{H}$ is a class of functions in a reproducing kernel Hilbert space (Gretton et al., 2012), $k(\cdot, \cdot)$ is the kernel function, and m, n are the number of samples for P and Q.

Metric 4. Wasserstein Distance

Arjovsky & Bottou (2017) propose the use of the dual representation of Wasserstein distance (Villani, 2009) for training GANs. The Wasserstein distance is an IPM which considers the 1-Lipschitz function class $\|F\|_L \leq 1$.

$$D_{\mathcal{W}}(P, Q) = \max_{\varphi} \left[\mathbb{E}_{x\ \sim p(X)}\left[D_\varphi(x)\right] - \mathbb{E}_{z\ \sim p(Z)}\left[D_\varphi(G_\theta(z))\right]\right] \tag{8}$$

Algorithm 1 Computing the divergence/distance

procedure DISTANCEOPTIMIZATIONPROCEDURE(Dataset $\{X_{tr}, X_{vl}, X_{te}\}$, generative model G_θ, learning rate η, metric D)
  Initialize critic network parameter φ.
  while φ has not converged **do**
    Sample data points from X, $\{x_m\} \sim X_{tr}$.
    Sample points from generative model, $\{s_m\} \sim G_\theta$.
    $\varphi \leftarrow \varphi + \eta \nabla_\varphi D(\{x_m\}, \{s_m\}; \varphi)$.
  Sample points from generative model, $\{s_m\} \sim G_\theta$.
  return $D(X_{te}, \{s_m\})$.

4 EXPERIMENTS

We conducted our experiments on three types of GANs: Deep Convolutional Generative Adversarial Networks (DCGAN), Least-Squares GAN (LS-GAN), and Wasserstein GAN (W-GAN). We evaluated these three families of GANs with five metrics. Two existing metrics, the inception score (IS) and MMD, were chosen because IS is the most commonly used metric for evaluating GANs, and MMD is well-known for two sample tests (Sutherland et al., 2017). The other three metrics have the same criterion as the standard GAN, LS-GAN, and W-GAN, with a fixed generator from section 3. In order to not confuse the test metric names with the actual trained GANs, we will denote the GAN, LS-GAN, and W-GAN metric at test time as JS, LS, and IW, respectively. Because the optimization of the discriminator is required both during training and test time, we will call the discriminator at test time *the critic*, in order to not confuse the two discriminators.

In our experiments, we considered the MNIST and CIFAR-10 datasets. MNIST consists of 60,000 training and 10,000 test images of size 28×28 pixels containing handwritten digits from the classes 0 to 9. From the 60,000 training examples, we set aside 10,000 as validation examples to tune various hyper-parameters. The CIFAR-10 dataset [1] consists of images with a size of $32 \times 32 \times 3$ pixels with ten different classes of objects. We used 45,000, 5,000 and 10,000 examples as training, validation, and test data, respectively. The hyper-parameters, such as learning rate, number of epochs, and batch size, were selected from discrete ranges and chosen based on a held-out validation set.

Table 8 in the supplementary materials (S.M.) section shows the learning rates and the convolutional kernel sizes that were used for each experiment. The architecture of each network is presented in the S.M. in Figure 10. Additionally, we used exponential-mean-square kernels with several different sigma values for MMD. A pre-trained logistic regression and pre-trained residual network are used for IS on MNIST and CIFAR-10 datasets, respectively. For every experiment, 10-fold evaluation was used[2] and the mean and standard deviation were displayed.

We analyze the GANs in the following way: First, we present the result of the GANs under the five different metrics. We then show the changes in performance with respect to the size of the architectures. Next, we investigate the performance with respect to a varying number of update ratios between discriminator and generator updates. Lastly, we investigate how performance changes with respect to the amount of training data available.

Notes on Existing Metrics

The log-likelihood measurement is the most commonly used metric for generative models. However, our experiments show that the result of measuring the log-likelihood using AIS[3] on GANs is strange, as shown in Figure 1. We measured the log-likelihood of DCGAN on MNIST with three different sizes of variance, 0.01, 0.025, and 0.05. The figure illustrates that the log-likelihood curve over the training epochs varies substantially depending on the variance, which indicates that the fixed

[1] `https://github.com/Lasagne/Recipes/blob/master/papers/deep_residual_learning/Deep_Residual_Learning_CIFAR-10.py`

[2] 10 fold refers to training ten GANs and computing the mean and standard deviation of the test distance, rather than using ten bags of samples from a single-trained GAN to compute the mean and standard deviation of the test distance.

[3] We used the original source code from `https://github.com/tonywu95/eval_gen`

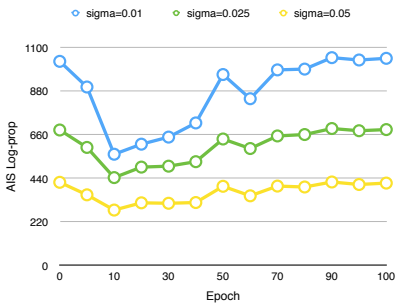
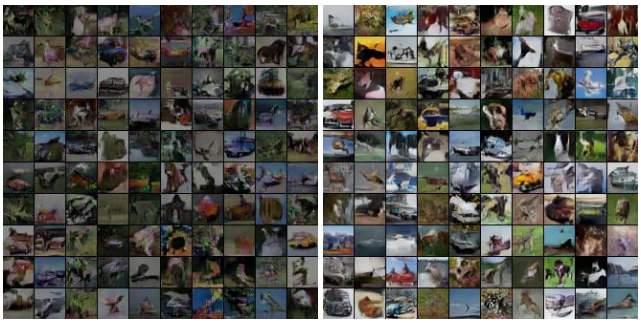


Figure 1: Log-likelihood using AIS during DCGAN training on MNIST.



(a) Inception Score = 6.45 (b) Inception Score = 6.31

Figure 2: Misleading examples of Inception Scores.

Gaussian observable model might not be the ideal assumption for GANs. Moreover, we observe a high log-likelihood at the beginning of training, followed by a drop in likelihood, which then returns to the high value, and we are unable to explain why this happens.

Among the metrics we considered, only IS and MMD were previously used to evaluate GANs. It was difficult to determine whether the IS and MMD matched with the proposed metrics. We observed that IS is consistent most of the time with the metrics that we proposed, and yet, we did find a few outlier cases when IS and MMD failed. For example, Figure 2 is an odd scenario, where the visual quality of samples did not match the score. The left side of the figure shows the samples of DCGAN when it failed to train properly. Even though the failed DCGAN samples are much darker than the samples on the right, the IS score for the left samples is higher than the right samples (a higher score is better for IS). Another odd example for MMD is shown in Figure 5. The samples on the left are dark, like the previous examples, but still textually recognizable, whereas the samples on the right are totally meaningless. However, MMD gives lower scores to the right side samples (a lower score is better for MMD). This is because the average intensity of the pixels of the samples on the left are closer to the training data than the samples on the right.

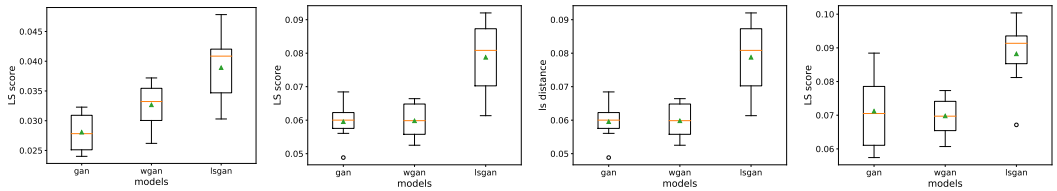
It seems that the common failure case of MMD is when the mean pixel intensities are a better match than texture matches (see Figure 5), and the common failure cases of IS happens to be when the samples are recognizable textures, but the intensity of the samples are either brighter or darker (see Figure 2). Other metrics probably have unusual failings as well, but they were not as obviously noticeable as those for IS and MMD. Thus, we conclude that IS and MMD scores do not always correlate with the sample quality.

Table 2: GAN scores under various metrics trained on MNIST. Lower scores are better for MMD and IW scores and vice versa for other scores.

Model	MMD	IW	JS	LS	IS (Logistic Reg.)
DCGAN	0.028 ± 0.0066	7.01 ± 1.63	2.2e-3 $\pm$ 3e-4	0.12 ± 0.013	5.76 ± 0.10
W-DCGAN	0.006 ± 0.0009	7.71 ± 1.89	4e-4 $\pm$ 4e-4	0.05 ± 0.008	5.17 ± 0.11
LS-DCGAN	0.012 ± 0.0036	4.50 ± 1.94	3e-3 $\pm$ 6e-4	0.13 ± 0.022	6.07 ± 0.08

Table 3: GAN scores under various metrics trained on CIFAR-10. Lower scores are better for MMD and IW scores and vice versa for other scores.

Model	MMD	IW	LS	IS (ResNet)
DCGAN	0.0538 ± 0.014	8.844 ± 2.87	0.0408 ± 0.0039	6.649 ± 0.068
W-DCGAN	0.0060 ± 0.001	9.875 ± 3.42	0.0421 ± 0.0054	6.524 ± 0.078
LS-DCGAN	0.0072 ± 0.0024	7.10 ± 2.05	0.0535 ± 0.0031	6.761 ± 0.069



(a) Filter # : [3, 64, 128, 256]

(b) Filter # : [3, 128, 256, 512]

(c) Filter # : [3, 256, 512, 1024]

(d) Filter #: [3, 320, 640, 1280]

Figure 3: GAN evaluation using different architectures for the critic (# of feature maps in each layer of the CNN critic). Figures (a,b,c) are evaluation under Wasserstein distance and Figures (d,e,f) are evaluation under least-square loss.

The metric comparison

Table 2 and Table 3 present the results on MNIST and CIFAR-10, respectively. Considering the proposed method of evaluation, we need to first ask whether the metric favours the samples from the family of model that was trained on the same metric; fortunately, we do not see this behaviour. Both for MNIST and CIFAR-10, all metrics (IW, JS, and LS) consistently agree with which samples are the closest to test data. We speculate that this happens for a combination of two reasons, where we are training the critic at test time from scratch, and the dynamics of the optimization process for training and test time are different.

Overall, every metric except for MMD showed that LS-DCGAN performed the best among the three GANs under our chosen hyper-parameter range, and that MMD favoured W-DCGAN's samples. The performance between DCGAN or W-DCGAN are quite similar, where the mean plus/minus standard deviation overlaps, especially for CIFAR-10. Moreover, we observed similar results for a range of different critic CNN architectures (number of feature maps in each convolutional layer): $[3, 64, 128, 256]$, $[3, 128, 256, 512]$, $[3, 256, 512, 1024]$, and $[3, 320, 640, 1280]$ (see Figure 3 and S.M. Figure 11).

More generally, because we see some overlap in the statistics of performance, the statistical significance becomes questionable. We computed the Wilcoxon rank sum test that the medians of the distributions are the same. The p-values in Table 4 indicate that the differences are significant for LS and not for IW.

Table 4: The P-values on CIFAR10 (Significant p < .05, two-tailed Wilcoxon rank sum test, N = 8)

	LS	IW
DCGAN:W-DCGAN	0.0013	0.6964
DCGAN:LS-DCGAN	0.0010	0.3822
LS-DCGAN:W-DCGAN	0.0266	0.6964

DCGANs are known to be unstable for a few reasons: the support of the data and model distributions being disjoint (Arjovsky & Bottou, 2017), and the Hessian of the GAN objective being non-Hermitian (Mescheder et al., 2017). LS-DCGAN and W-DCGAN are proposed to address this by providing non-saturating gradients. Based on experience, it is found that LS-DCGAN is the simplest and most stable model to train. We visualize the 2D subspace of the loss surface of the GANs in S.M. Figure 21. Here, we took the parameters of three trained models (corresponds to red vertices in the figure) and applied barycentric interpolation with respect to three parameters (see details from (Im et al., 2016c)). It appears that DCGAN surfaces have much sharper slopes when compared to LS-DCGAN and W-DCGAN[4], and LS-DCGAN has the most gentle surfaces. This geometric view is consistent with our finding that LS-DCGAN is the easiest and the most stable to train.

Table 5: Reference for the different architectures explored in the experiments.

Label	Feature Maps	
	Discriminator	Generator
(a)	[3, 16 , 32 , 64]	[128 , 64 , 32 , 3]
(b)	[3, 32 , 64 , 128]	[256 , 128, 64 , 3]
(c)	[3, 64 , 128, 256]	[512 , 256, 128, 3]
(d)	[3, 128, 256, 512]	[1024, 512, 256, 3]
(e)	[3, 16 , 32 , 64]	[1024, 512, 256, 3]
(f)	[3, 128, 256, 512]	[128 , 64 , 32 , 3]

[4]The local zig-zag patterns in the figures are minor artifacts from rendering.

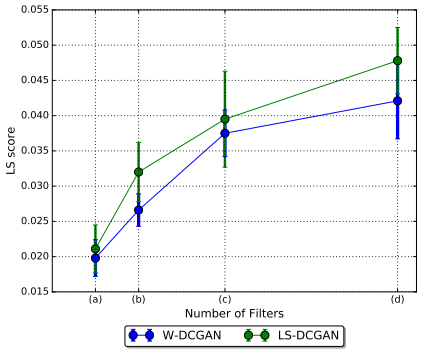


Figure 4: LS score evaluation of W-DCGAN & LS-DCGAN w.r.t number of feature maps.

(a) Samples from (e) in Table 5, MMD= 0.03, IS= 5.11

(b) Samples from (f) in Table 5, MMD= 0.49, IS= 6.15

Figure 5: W-DCGAN trained with different numbers of feature maps.

Table 6: LS-DCGAN and W-DCGAN scores on CIFAR-10 with respect to different generator and discriminator capacity.

Model	Architecture (Table 5)	MMD Test vs. Samples	IW	LS	IS (ResNet)
W-DCGAN	(e)	0.1057 ± 0.0798	450.17 ± 25.74	0.0079 ± 0.0009	6.403 ± 0.839
	(f)	0.2176 ± 0.2706	16.52 ± 15.63	0.0636 ± 0.0101	6.266 ± 0.055
LS-DCGAN	(e)	0.1390 ± 0.1525	343.23 ± 47.55	0.0092 ± 0.0007	5.751 ± 0.511
	(f)	0.0054 ± 0.0022	12.75 ± 4.29	0.0372 ± 0.0068	6.600 ± 0.061

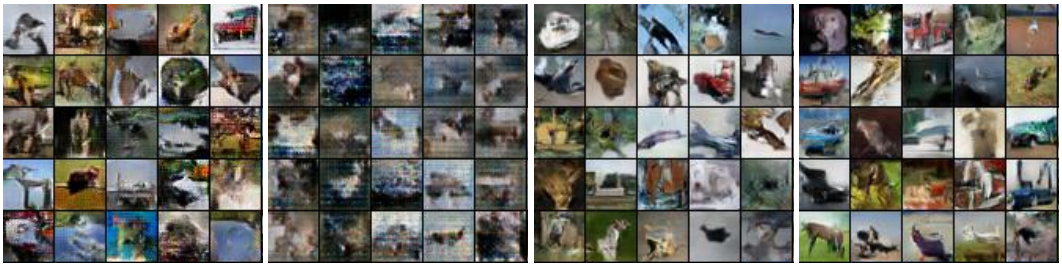
Performance change with respect to the size of the network

Several works have demonstrated an improvement in performance by enlarging deep network architectures (Krizhevsky et al., 2012; Simonyan & Zisserman, 2014; He et al., 2015; Huang et al., 2017). Here, we investigate performance changes with respect to the width and depth of the networks.

First, we trained three GANs with different numbers of feature map sizes, as shown in Table 5 (a-d). Note that we double the number of feature maps in Table 5 for both the discriminators and generators. In Figure 4, the performance of the LS-score increases logarithmically as the number of feature maps is doubled. A similar behaviour is observed in other metrics as well (see S.M. Figure 12).

We then tested the importance of size in the discriminative versus generative network. We considered two extreme feature map sizes, where we choose a small and a large number of feature maps for the generator and discriminator, and vice versa (see Label (e) and (f) in Table 5), and results are shown in Table 6. For LS-DCGAN, it can be seen that the large number of feature maps for the discriminator has a better score than the large number of feature maps for the generator. This can also be qualitatively verified by looking at the samples from architectures (a), (e), (f), and (d) in Figure 6. For W-DCGAN, we observe the agreement between the LS and IW metric, but conflicts with MMD and IS. But, when we look at the samples of W-DCGAN in Figure 5, it is clear that the model with the larger number of feature maps in the discriminator should get a better score. This is another example of false intuition propagated by MMD and IS. One interesting observation is that when we compare the score and samples from architecture (a) and (e) from Table 5, architecture (a) is much better than (e) (see Figure 6). It demonstrates that having a large generator and small discriminator is worse than having small architectures for both networks. Overall, we found that having a larger generator than a discriminator does not give good results, and that it is more desirable to have a larger discriminator than a generator. Similar results were also observed for MNIST, as shown in S.M. Figure 16. This result somewhat supports the theoretical result from Arora et al. (2017), where the generator capacity needs to be modulated in order for approximately pure equilibrium to exist for GANs.

Lastly, we experimented with how performance changes with respect to the dimension of the noise vectors. The source of the sample starts by transforming a noise vector into a meaningful image. It is unclear how the size of noise affects the ability of the generator to generate a meaningful image. Che et al. (2017) have observed that a 100-d noise vector preserves modes better than a 200-d noise vector



"Small" number of filters for both discriminator and generator (ref. (a) in Table 5).

"Small" and "large" number of filters for discriminator and generator respectively (ref. (e) in Table 5).

"Large" and "small" number of filters for discriminator and generator respectively (ref. (f) in Table 5).

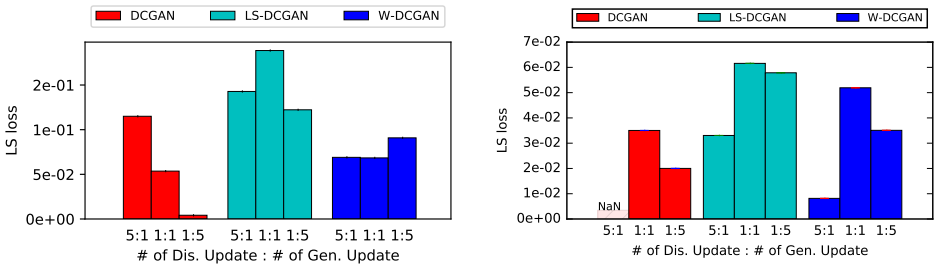
"Large" number of filters for both discriminator and generator (ref. (d) in Table 5).

Figure 6: Samples from different LS-DCGAN architectures.

for DCGAN. Our experiments show that this depends on the model. Given a fixed size architecture (d) from Table 5, we observed the performance of LS-DCGAN and W-DCGAN by varying the size of noise vector z. Table 7 illustrates that LS-DCGAN gives the best score with a noise dimension of 50 and W-DCGAN gives best score with a noise dimension of 150 for both IW and LS. The outcome of LS-DCGAN is consistent with the result in (Che et al., 2017). It is possible that this is because both fall into the category of f-divergences, whereas the W-DCGAN behaves differently because its metric falls under a different category, the Integral Probability Metric.

Table 7: LS-DCGAN and W-DCGAN scores on CIFAR-10 with respect to the dimensionality of the noise vector.

|z|	LS-DCGAN		W-DCGAN	
	IW	LS	IW	LS
50	3.9010 ± 0.60	0.0547 ± 0.0059	6.0948 ± 3.21	0.0532 ± 0.0069
100	5.6588 ± 1.47	0.0511 ± 0.0065	5.7358 ± 3.25	0.0528 ± 0.0051
150	5.8350 ± 0.80	0.0434 ± 0.0036	3.6945 ± 1.33	0.0521 ± 0.0050



(a) MNIST

(b) CIFAR10

Figure 7: LS score evaluation with respect to a varying number of discriminator and generator updates on DCGAN, W-DCGAN, and LS-DCGAN.

Performance change with respect to the ratio of number of updates between the generator and discriminator

In practice, we alternate the updates between the discriminator and generator, and yet this is not guaranteed to give the same result as the solution to the min-max problem in Equation 2. Hence, the update ratio can influence the performance of GANs. We experimented with three different update ratios, $5:1$, $1:1$, and $1:5$, with respect to the discriminator and generator update. We applied these ratios to both the MNIST and CIFAR-10 datasets on all models.

Figure 7 presents the LS scores on both MNIST and CIFAR-10 and this result is consistent under the IW metric as well (see S.M. Figure 17). However, we did not find any pattern of one update ratio

being superior over others among the two datasets. For CIFAR-10, $1:1$ update ratio worked the best for all models, and yet for MNIST, different ratios worked the best for different models. Hence, we conclude that number of update ratio needs to be dynamically tuned. The corresponding samples from the different models trained by different update ratios are shown in S.M. Figure 19.

Performance with respect to the amount of available training data

In practice, DCGANs are known to be unstable, and the generator suffers as the discriminator gets better, due to disjoint support between the data and generator distributions (Goodfellow et al., 2014; Arjovsky & Bottou, 2017). W-DCGAN and LS-DCGAN offer alternative ways to solving this problem in GANs. If the model is suffering from disjoint support, having more training examples will not help, and alternatively, if the model does not suffer from such a problem, having more training examples could potentially help. Here, we explore how sensitive three different kinds of GANs are with respect to the number of training examples. We have trained GANs with 10,000, 20,000, 30,000, 40,000, and 45,000 examples on CIFAR-10. Figure 8 shows that the LS score curve of DCGAN grows quite slowly when compared to W-DCGAN and LS-DCGAN. The three GANs have relatively the same loss when they are trained with 10,000 training examples. However, the DCGAN only gained 0.0124 ± 0.00127 by increasing from 10,000 to 40,000 training examples, whereas the performance of W-DCGAN and LS-DCGAN improved by 0.03016 ± 0.00469 and 0.0444 ± 0.0033, respectively. Thus, we empirically observe that W-DCGAN and LS-DCGAN increases their performance faster than DCGAN as the number of training examples grows.

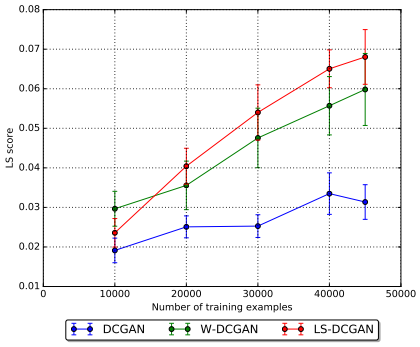


Figure 8: LS score evaluation on W-DCGAN & LS-DCGAN w.r.t number of data points.

5 Conclusion

In this paper, we proposed four well-known distance functions as an evaluation metrics, and empirically investigated the DCGAN, W-DCGAN, and LS-DCGAN families under these metrics. Previously, these models were compared based visual assessment of sample quality and difficulty of training. In our experiments, we showed that there are performance differences in terms of average experiments, but that some are not statistically significant. Moreover, we thoroughly analyzed the performance of GANs under different hyper-parameter settings.

There still are many different type of GANs that need to be evaluated, such as GRAN (Im et al., 2016a), IWGAN (Gulrajani et al., 2017), BEGAN (Berthelot et al., 2017), MMDGAN (Li et al., 2017), and CramerGAN (Bellemare et al., 2017). We hope to evaluate all of these models under this framework and thoroughly analyze them in the future. Moreover, there has been work on taking ensemble approaches to GANs, such as Generative Adversarial Parallelization Im et al. (2016b). Ensemble approaches have been empirically shown to work well in many domains of research, so it would be interesting to find out whether ensembles can also help in min-max problems. Alternatively, we can also try to evaluate other log-likelihood-based models like NVIL (Mnih & Gregor, 2014), VAE (Kingma & Welling, 2014), DVAE (Im et al., 2015), DRAW (Gregor et al., 2015), RBMs (Hinton et al., 2006; Salakhutdinov & Hinton, 2009), NICE Dinh et al. (2014), etc.

Model evaluation is an important and complex topic. Model selection, model design, and even research directions, can change depending on the evaluation metric. Thus, we need to continuously explore different metrics and rigorously evaluate new models.

REFERENCES

Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. In *arXiv preprint arXiv:1701.04862*, 2017.

Martin Arjovsky, Soumith Chintala, and Leon Bottou. Wgan. In *arXiv preprint arXiv:1701.07875*, 2017.

Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. Generalization and equilibrium in generative adversarial nets. In *arXiv preprint arXiv:1703.00573*, 2017.

Marc G. Bellemare, Ivo Danihelka, Will Dabney, Shakir Mohamed, Balaji Lakshminarayanan, Stephan Hoyer, and Rémi Munos. The cramer distance as a solution to biased wasserstein gradients. In *arXiv preprint arXiv:1705.10743*, 2017.

David Berthelot, Thomas Schumm, and Luke Metz. Began: Boundary equilibrium generative adversarial networks. In *arXiv preprint arXiv:1703.10717*, 2017.

Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. Mode regularized generative adversarial networks. In *arXiv preprint arXiv:1705.08584*, 2017.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: non-linear independent components estimation. In *arXiv preprint arXiv:1410.8516*, 2014.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair†, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the Neural Information Processing Systems (NIPS)*, 2014.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Scholkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13:723–773, 2012.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. In *arXiv preprint arXiv:1704.00028*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *arXiv preprint arXiv:1512.03385*, 2015.

Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the Computer Vision and Pattern Recognition (CVPR)*, 2017.

Daniel Jiwoong Im, Sungjin Ahn, Roland Memisevic, and Yoshua Bengio. Denoising criterion for variational auto-encoding framework. In *arXiv preprint arXiv:1509.00519*, 2015.

Daniel Jiwoong Im, Dongjoo Kim, Hui Jiang, and Roland Memisevic. Generating images with recurrent adversarial networks. In *arXiv preprint arXiv:1602.05110*, 2016a.

Daniel Jiwoong Im, He Ma, Dongjoo Kim, and Graham Taylor. Generating adversarial paralleliation. In *arXiv preprint arXiv:1612.04021*, 2016b.

Daniel Jiwoong Im, Michael Tao, and Kristin Branson. An empirical analysis of the optimization of deep network loss surfaces. In *arXiv preprint arXiv:1612.04010*, 2016c.

Diederik P Kingma and Max Welling. Auto-encoding varational bayes. In *Proceedings of the Neural Information Processing Systems (NIPS)*, 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the Neural Information Processing Systems (NIPS)*, 2012.

Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. Mmd gan: Towards deeper understanding of moment matching network. In *arXiv preprint arXiv:1705.08584*, 2017.

Xudong Mao, Qing Li, Haoran Xie, Raymond Y.K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *arXiv preprint arXiv:1611.04076*, 2017.

Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. The numerics of gans. In *Proceedings of the Neural Information Processing Systems (NIPS)*, 2017.

Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2014.

Alfred Muller. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29(2):429–443, 1997.

XuanLong Nguyen, Martin J. Wainwright, and Michael I. Jordan. Estimating divergence functionals and the likelihood ratio by penalized convex risk minimization. In *Proceedings of the Neural Information Processing Systems (NIPS)*, 2008.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *arXiv preprint arXiv:1606.00709*, 2016.

Ruslan Salakhutdinov and Geoffrey E. Hinton. Deep boltzmann machines. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2009.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *arXiv preprint arXiv:1606.03498*, 2016.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *arXiv preprint arXiv:1409.1556*, 2014.

Bharath Sriperumbudur, Kenji Fukumizu, Arthur Gretton, Bernhard Schölkopf, and Gert Lanckriet. On integral probability metrics, phi-divergences and binary classification. 01 2009.

Dougal J. Sutherland, Hsiao-Yu Tung, Heiko Strathmann, Soumyajit De Aaditya Ramdas, Alex Smola, and Arthur Gretton. Generative models and model criticism via optimized maximum mean discrepancy. In *arXiv preprint arXiv:1611.04488*, 2017.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *arXiv preprint arXiv:1512.00567*, 2015.

Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. 5 November 2015.

Cedric Villani. Grundlehren der mathematischen wissenschaften. In *Optimal Transport: Old and New*. Springer, Berline, 2009.

Yuhuai Wu, Yuri Burda, Ruslan Salakhutdinov, and Roger Grosse. On the quantitative analysis of decoder based generative models. In *International Conference on Learning Representation*, 2016.

6 MODEL ARCHITECTURE

Table 8: Hyper-parameters used for different experiments.

		GAN training			Critic Training (test time)		
	Model	Disc. Lr.	Gen. Lr.	Ratio[5]	Cr. Lr.	Cr. Kern	Num Epoch
Table 2	DCGAN	0.0002	0.0004	1:2	0.0001	[1, 128, 32]	25
	W-DCGAN	0.0004	0.0008	1:1			
	LS-DCGAN	0.0004	0.0008	1:2			
Table 3	DCGAN	0.0002	0.0001	1:2	0.0002	[3, 128, 256, 512]	11
	W-DCGAN	0.0008	0.0004	1:1			
	LS-DCGAN	0.0008	0.0004	1:2			
Table 6	DCGAN	0.0002	0.0001	1:2	0.0002	[3, 128, 256, 512]	11
	W-DCGAN	0.0002	0.0001	1:1			
	LS-DCGAN	0.0008	0.0004	1:2			
Figure 7b	DCGAN	0.0001	0.00005	5:1	0.0002	[3, 128, 256, 512]	11
				1:1			
				1:5			
	W-DCGAN	0.0008	0.0004	5:1			
				1:1			
				1:5			
	LS-DCGAN			5:1			
				1:1			
				1:5			
Figure 18	DCGAN	0.0001	0.00005	5:1	0.0002	[1, 128, 32]	25
				1:1			
				1:5			
	W-DCGAN	0.0008	0.0004	5:1			
				1:1			
				1:5			
	LS-DCGAN			5:1			
				1:1			
				1:5			
Figure 12	DCGAN	0.0002	0.0001	1:2	0.0002	[3, 128, 256, 512]	11
	W-DCGAN	0.0002	0.0001	1:1			
	LS-DCGAN	0.0008	0.0004	1:2			
Figure 20	DCGAN	0.0002	0.0001	1:5	0.0002	[3, 256, 512, 1028]	11
	W-DCGAN	0.0008	0.0004	1:1			
	LS-DCGAN	0.0008	0.0004	1:5			

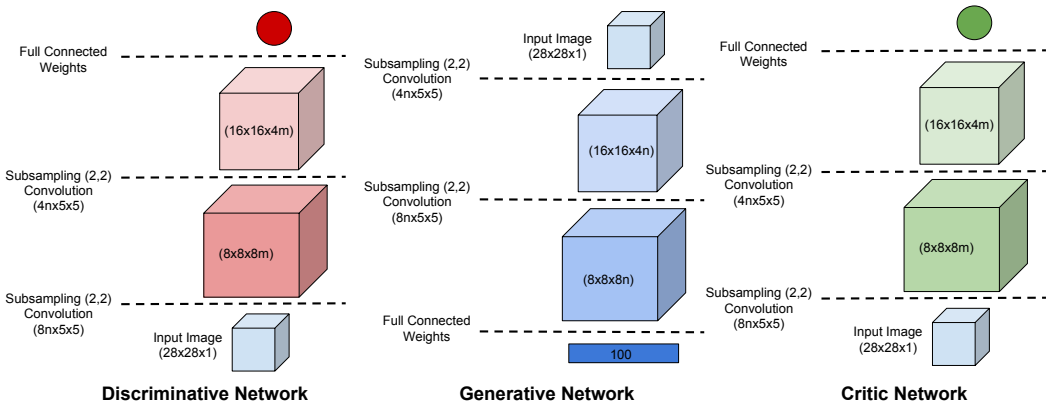


Figure 9: GAN Topology for MNIST.

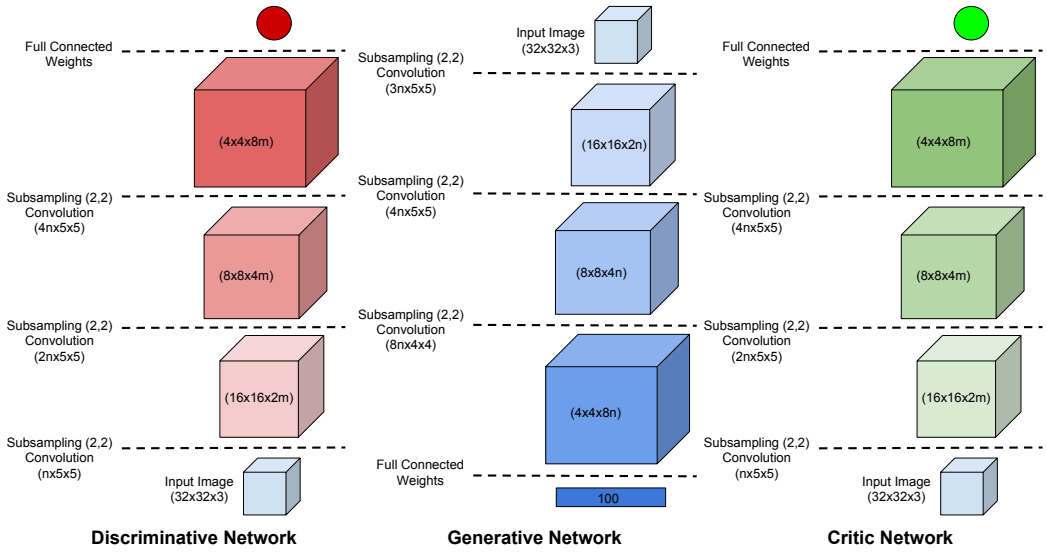
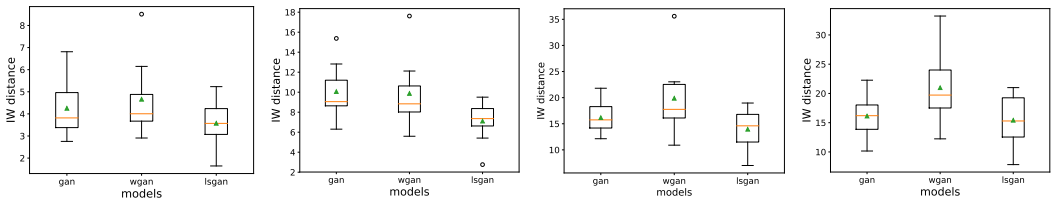


Figure 10: GAN Topology for CIFAR10.



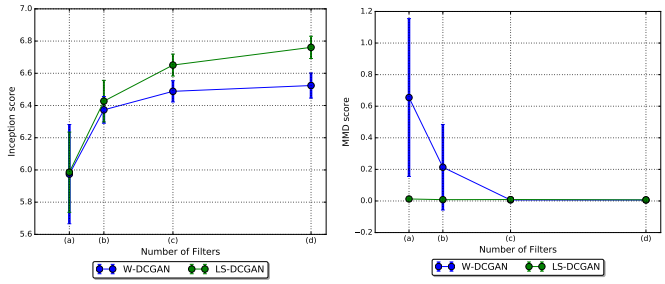
(a) Filter # : [3, 64, 128, 256]

(b) Filter # : [3, 128, 256, 512]

(c) Filter # : [3, 256, 512, 1024]

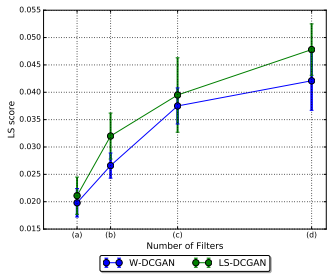
(d) Filter # : [3, 320, 640, 1280]

Figure 11: GAN evaluation using different critic's architecture (# of filter of critic's convolutional network). Figure (a,b,c) are evaluation under Wasserstein distance.

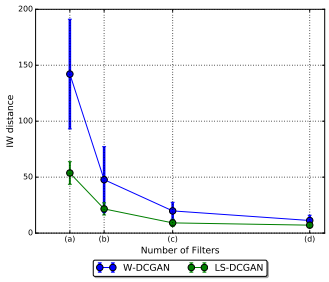


(a) IS (the higher the better)

(b) MMD (the lower the better)



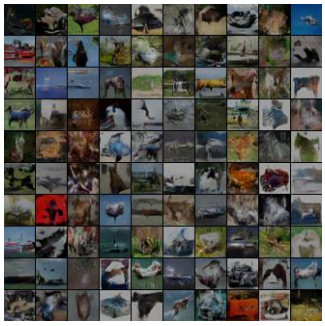
(c) LS (the higher the better)



(d) IW (the lower the better)

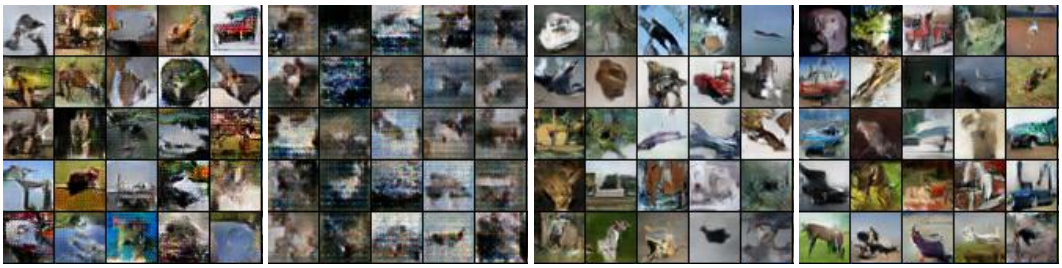
Figure 12: Performance of W-DCGAN & LS-DCGAN with respect to number of filters.

(a) Samples from (e) in Table 5, MMD= 0.03, IS= 5.11



(b) Samples from (f) in Table 5, , MMD= 0.49, IS= 6.15

Figure 13: W-DCGAN trained with different number of filters.



(a) “Small” number of filters for both discriminator and generator (ref. (a) in Table 5).

(b) “Small” and “large” number of filters for discriminator and generator respectively (ref. (e) in Table 5).

(c) “Large” and “small” number of filters for discriminator and generator respectively (ref. (f) in Table 5).

(d) “Large” number of filters for both discriminator and generator (ref. (d) in Table 5).

Figure 14: Samples from different architectures of LS-DCGAN

(a) “Small” number of filters for both discriminator and generator (ref. (a) in Table 5).

(b) “Small” and “large” number of filters for discriminator and generator respectively (ref. (e) in Table 5).

(c) “Large” and “small” number of filters for discriminator and generator respectively (ref. (f) in Table 5).

(d) “Large” number of filters for both discriminator and generator (ref. (d) in Table 5).

Figure 15: Samples from different architectures of W-DCGAN.

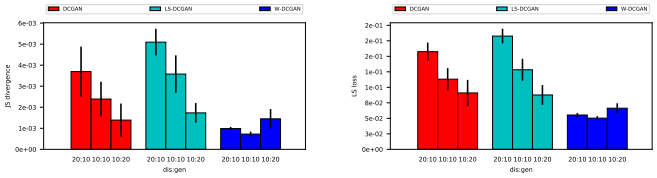
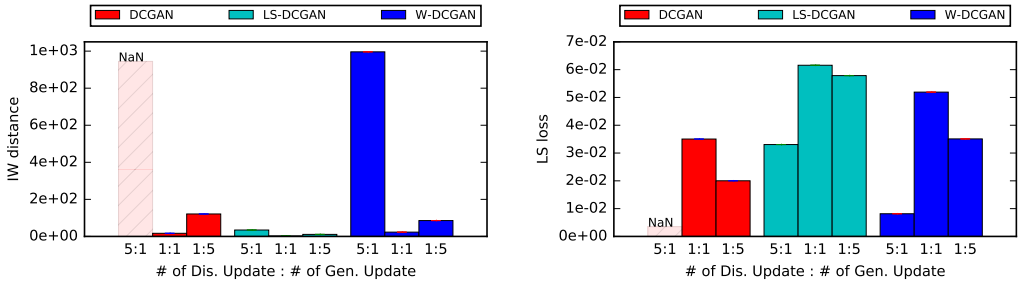


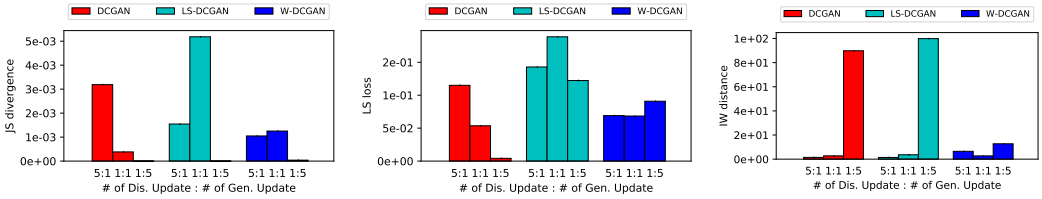
Figure 16: The performance of GANs trained with different numbers of feature maps.



(a) IW distance (The lower the better)

(b) LS divergence (The higher the better)

Figure 17: Performance over DCGAN, W-DCGAN, and LS-DCGAN trained with different number of discriminator and generator updates. These models were trained on CIFAR10 dataset and evaluated on IW and LS metric.



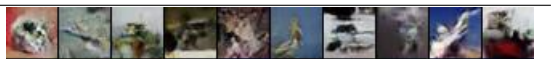
(a) JS divergence (The higher the better)

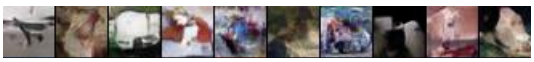
(b) LS divergence (The higher the better)

(c) IW distance (The lower the better)

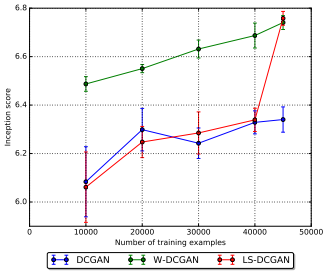
Figure 18: Performance over DCGAN, W-DCGAN, and LS-DCGAN trained with different number of discriminator and generator updates. These models were trained on MNIST dataset and evaluated on JS, LS, and IW metric.

Figure 19: Samples at different numbers of update ratios.

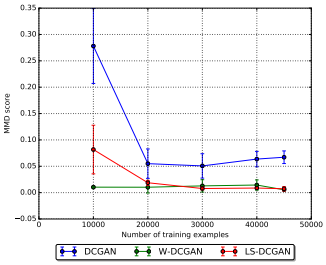
Ratio	DCGAN Samples
1:1	
1:5	

Ratio	W-DCGAN Samples
5:1	
1:1	
1:5	

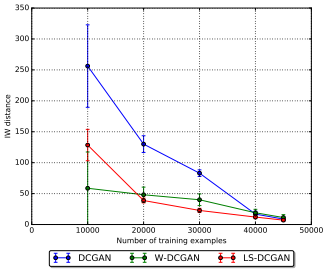
Ratio	LS-DCGAN Samples
5:1	
1:1	
1:5	



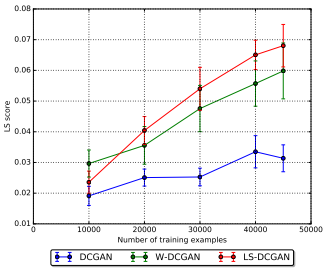
(a) IS Score (the lower the better)



(b) MMD Score (the higher the better)

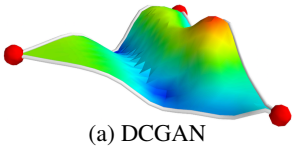


(c) IW Score (the lower the better)

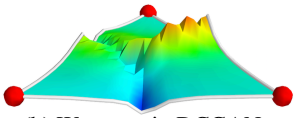


(d) LS Score (the higher the better)

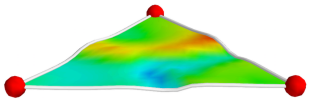
Figure 20: Performance of W-DCGAN & LS-DCGAN with respect to number of data points



(a) DCGAN



(b) Wasserstein DCGAN



(c) Least-Square DCGAN

Figure 21: Interpolation between three final GAN parameters trained using different random seeds on CIFAR10. Loss surface values are amplified by 10 times in order to illustrate the separation of the terrains.