Evaluation of Spectral Normalization for GANs Using Inception Score

Eysteinn Gunnlaugsson, Charles Hamesse, Egill Vignisson

Group 16

Abstract. We evaluate the effect of spectral normalization on various Deep Convolutional Generative Adversarial Network (DCGAN) configurations. We start with a vanilla DCGAN and then consider the Wasserstein loss and several variants. We evaluate our models on CIFAR10 and a dataset that we collected ourselves consisting of animal pictures, using the Inception score as performance indicator. Experimental results show that spectral normalization helps making the training of GANs more stable and improves the quality of the generated images.

Keywords: Deep generative models, Generative adversarial networks, Image generation

1 Introduction

Since their introduction in 2014, Generative Adversarial Networks (GANs) [1] have gained a lot of popularity in the machine learning community and more specifically in the field of generative models. The GAN framework has been applied with admirable results to many different applications such as computer vision [2] [3] [4], speech synthesis [5] [6], or drug discovery [7].

We give a brief overview of the GAN framework. GANs consist of two major components, namely a generator and a discriminator, both consisting of a neural network. The aim of the generator is to generate samples that mimic a real data distribution from noise, and the aim of the discriminator is to distinguish samples that come from the real data distribution from the ones produced by the generator. Trained consecutively as an adversarial pair, these networks will enter a game in which the generator is encouraged to model the real data distribution as closely as possible with the discriminator acting as a critic as accurate as possible. On top of their practical abilities, this makes GANs very interesting for their inherent theoretical aspects, paving the way to leveraging methods and techniques that were formerly used in game theory exclusively.

Now, the training of GANs is an infamously intricate task. In fact, a significant amount of the research on GANs has been focusing on how to improve the stability of their training. More specifically, the performance of the discriminator gets hard to control since the density ratio estimation in high-dimensional spaces is commonly inaccurate and unstable; the generator can then hardly learn the multimodal structure of the target distribution.

There has been numerous efforts to overcome this stability issue, amongst which expressing the network's objective functions differently, i.e. re-writing the discriminator and generator losses was shown promising. For instance, a different probability distribution divergence measure is proposed with the Wasserstein GAN (WGAN) [8]. Another example is the Least Squares GAN (LSGAN), where the vanishing gradient problem of the sigmoid function used in the original GAN is exposed and a solution is proposed by adopting a least squares loss function for the discriminator loss.

In a different approach, more architectural this time, spectral normalization proposes a method that reduces the function space from which can originate the discriminator to the space of Lipschitz continuous functions, assuring the boundedness of statistics [9]. This is done by normalizing the weight matrix of each layer using their spectral norm. The authors show that doing so not only improves the stability of the learning, but also that the method is easily implemented and tuned. We describe spectral normalization in more details in Section 2.5. This recent development has received a hugely positive appraisal and is seen by many as a major breakthrough in deep learning.

In this work, we implement a number of different GANs for image generation and add spectral normalization [10] to evaluate its effect on all of our configurations. For instance, we try WGAN [8], which works under the assumption that the discriminator function is Lipschitz continuous; a property assured by spectral normalization. In this regard, it makes sense to run experiments on WGAN with spectral normalization or different ways of ensuring the discriminator's Lipschitz continuity. In order to evaluate and compare the performance of our GANs, we use the Inception score [11]. We also discuss the relevance of this metric. As for the datasets, we use CIFAR10 [12] and another dataset that we gathered ourselves for this study consisting of 20K animal pictures (mostly reptiles), fetched from the Flickr API. In the following, we will denote it the "Reptiles" dataset. The purpose is to evaluate these techniques on arbitrary data and not the usual computer vision datasets.

2 Background

We present the framework of GANs starting with the original paper by Ian Goodfellow [1] then describe more recent advances in a chronological order, as depicted in Figure 1.

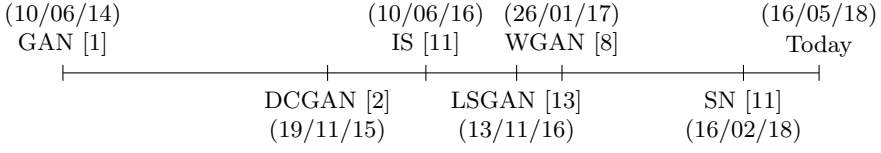


Fig. 1: Major contributions in the field of Generative Adversarial Networks.

In this work, we consider all of the techniques mentioned in Figure 1 except LSGAN [1].

2.1 Generative Adversarial Networks

Generative Adversarial Networks (GANs) were first introduced in [1]. They are a class of generative models trained in a adversarial manner by opposing two networks: a generative network G that learns the data distribution and a discriminative network D that tries and estimates the probability that a given sample came from the real training data rather than a generation from G.

The objective for G is to maximize the probability of D making a mistake, and the objective for D is to minimize that same probability. This framework corresponds to a minimax two-player game. In the space of arbitrary functions G and D, a unique equilibrium solution exists, with G recovering the real data distribution and D equal to 0.5 everywhere, meaning it's unable to distinguish the real images from the ones generated by G. Since G and D are (de-)convolutional networks, both can be trained using available backpropagation techniques.

To learn the distribution p_g over the real data $\boldsymbol{x}$, G starts from sampling input variables $\boldsymbol{z}$ from a distribution of our choice $p_z(\boldsymbol{z})$, then maps the input variables $\boldsymbol{z}$ to space $G(\boldsymbol{z};\theta_g)$ that should, after training, resemble the training data space. The discriminator, D, maps images to a boolean $D(\boldsymbol{x};\theta_d)$ indicating whether images are from training data or generated from G. The orignal minimax objective for GANs is defined as:

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

2.2 Deep Convolutional Generative Adversarial Networks

Building upon Goodfellow's work [1], Radford et al. apply the GAN framework to computer vision, bridging the gap between the success of Convolutional Neural Networks (CNNs) for supervised learning and GANs for unsupervised learning [2]. Training on various image datasets, authors show that the adversarial pair learns a hierarchy of features in both the generator and discriminator.

2.3 Inception Score

The Inception Score (IS) is first presented in [11]. This work (originating from OpenAI, whose team includes author of the original GAN paper Ian Goodfellow) presents a variety of new architectural features and training procedures meant to improve the training of GANs. Amongst other things, authors make the observation that GANs lack an objective function, making it difficult to compare the performance of different models. In the context of image generation, an intuitive

[1] As we progressed in our work, we figured out it was more relevant to investigate further with WGAN-related techniques than implementing a broad spectrum of techniques without really going in-depth.

performance metric can be obtained by human annotators assessing the quality of the generated images. However, this method isn't as scalable as one would wish for obvious reasons. The inception score is proposed as an alternative, and is shown to correlate well with human evaluation.

We describe the method briefly. By applying the Inception model [14] to all generated images, we get the conditional label distribution $p(y|\boldsymbol{x})$. Images that contain meaningful objects should have a conditional label distribution $p(y|\boldsymbol{x})$ with low entropy. Moreover, we expect the model to generate varied images, so the marginal $\int p(y|\boldsymbol{x} = G(z))dz$ should have high entropy. By combining these two requirements, the proposed metric becomes:

$$\mathrm{IS}(G) = \exp\left(\mathbb{E}_{\boldsymbol{x}\sim p_{\mathrm{data}}(\boldsymbol{x})} D_{\mathrm{KL}}(p(y|\boldsymbol{x})||p(y))\right). \tag{2}$$

Here D_{KL} is known as the Kullback-Leibler divergence and it measures how one probability distribution diverges from another by using a logarithmic difference:

$$D_{\mathrm{KL}}(P\|Q) = \int_{-\infty}^{\infty} p(x)\, \log \frac{p(x)}{q(x)}\, dx, \tag{3}$$

where $p(y|\boldsymbol{x})$ is the conditional label distribution while $p(y)$ is the marginalized label distribution, i.e. $p(y) = \int_{\boldsymbol{x}} (p(y|\boldsymbol{x})p_g(\boldsymbol{x})d\boldsymbol{x}$. In order to calculate the score we replace $p(y)$ with the empirical marginal class distribution $\hat{p}(y) = \sum_{i=1}^{N} p(y|\boldsymbol{x}^i)$ and estimate the expectation using the basic Monte Carlo estimator yielding:

$$\mathrm{IS}(G) = \exp\left(\sum_{i=1}^{N} D_{\mathrm{KL}}(p(y|\boldsymbol{x}^i)||\hat{p}(y))\right). \tag{4}$$

2.4 WGAN

The Wasserstein GAN (WGAN) algorithm is proposed as a consequence to various observations on how we usually measure the distance between the model distribution and the real distribution. In fact, the choice of divergence functions has a huge impact on how the discriminator will manage to classify samples as real or fake, and in turn how the generator will be led to adjust its parameters to mimic the real distribution. Authors show how the Earth Mover (EM) distance behaves in comparison to popular probability distances and divergences used in the context of learning distributions. WGANs are developed by making GANs minimize a reasonable and efficient approximation of the EM distance, working under the assumption that the discriminator functions are K-Lipschitz continuous. For a parameterized family of functions $\{f_w\}_{w\in\mathcal{W}}$ that all have the K-Lipschitz continuity property, the discriminator objective is defined as:

$$\max_{w\in\mathcal{W}} \mathbb{E}_{x\sim p_{\mathrm{data}}(x)}[f_w(x)] - \mathbb{E}_{z\sim p(z)}[f_w(g_\theta(z)]. \tag{5}$$

As a result, one of the most compelling practical benefits of WGANs is the ability to continuously estimate the EM distance by training the discriminator to optimality. Various ways to ensure Lipschitz continuity are described, including gradient penalty or weight clipping. For the latter, however, author note that it is "clearly terrible way to enforce a Lipschitz constraint" [8].

2.5 Spectral Normalization

Spectral normalization [10] was introduced in an effort to try and make the training of GANs more stable and the original paper received very positive criticism. [2]. As we noted in the previous section, Lipschitz continuity is a property of interest for GANs. More specifically, the assumption for the loss of Wasserstein GANs is that the functions yielded by the discriminator are K-Lipschitz continuous for some constant K. Spectral normalization aims to control the Lipschitz constant of this discriminator function f by constraining the spectral norm of each layer $g_l : \boldsymbol{h}_{l-1} \mapsto W^l \boldsymbol{h}_{l-1}$. We briefly review the method, but the interested reader will find a more comprehensive explanation in the original paper.

By definition, the Lipschitz norm $\|g\|_{\text{Lip}}$ is equal to $\sup_{\boldsymbol{h}} \sigma(\nabla g(\boldsymbol{h}))$, with $\sigma(A)$ being the spectral norm of the matrix A (which is equivalent to the largest singular value of A). Therefore, for a linear layer g_l, the norm is given by $\|g_l\|_{\text{Lip}} = \sup_{\boldsymbol{h}} \sigma(\nabla g_l(\boldsymbol{h})) = \sup_{\boldsymbol{h}} \sigma(W_l) = \sigma(W_l)$. Now, if the Lipschitz norm of the activation function $\|a_l\|_{\text{Lip}}$ is equal to 1 (a condition that ReLU or Leaky ReLU satisfy [15]), we can use the composition inequality $\|g_1 \circ g_2\|_{\text{Lip}} \leq \|g_1\|_{\text{Lip}} \cdot \|g_2\|_{\text{Lip}}$ and define the bound on $\|f\|_{\text{Lip}}$:

$$\|f\|_{\text{Lip}} \leq \prod_{l=1}^{L+1} \|a_l\|_{\text{Lip}} \|g_l\|_{\text{Lip}} = \prod_{l=1}^{L+1} \sigma(W^l). \tag{6}$$

For each layer, spectral normalization alters the weight matrix W_l so that it satisfies the Lipschitz constraint $\sigma(W_l) = 1$:

$$\bar{W}_{l\text{SN}}(W_l) := W_l / \sigma(W_l), \tag{7}$$

enforcing the upper bound of 1 for $\|f\|_{\text{Lip}}$ using the inequation 6.

3 Approach

For our research we replicated the DCGAN network, described in great detail in [2], in Tensorflow using Taehoon Kim's (found here) implementation as inspiration.

[2] As a matter of fact, this has been posted on the peer-reviewing platform:

> *"This is a great paper! I don't think this paper explains the importance of its results nearly enough and I'm concerned that it may not be obvious what a breakthrough it is just from skimming the abstract."* - Ian Goodfellow

3.1 Architecture

Generator architecture details. The generator's architecture is depicted in Figure 2. The input layer is a vector of size 100 followed by 4 convolutional hidden layers. The output layer is then the picture generated that feeds into the discriminator. The 4 hidden layers use the ReLU activation function while the output layer uses the tanh function. The authors of DCGAN state that using a bounded activation like tanh for the output layer allowed the network to *"learn more quickly to saturate and cover the color space of the training distribution"* [2] (p. 3).

Discriminator architecture details. The discriminator's architecture follows a very similar structure with the obvious exception of the input and output layers. The input layer for the discriminator consists of an image, while the output layer is a single neuron activated by a *sigmoid* function that represents the classification of real vs fake images. The convolutional layers use the leaky ReLU activation function.

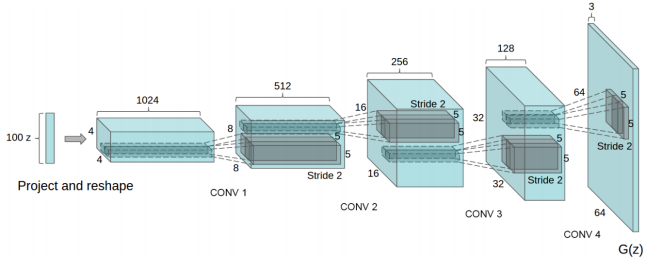


Fig. 2: Generator Network architecture. Cited from [2]

3.2 Training

We train all networks using the Adam optimizer [16] with a learning rate of 0.002 and a β_1 value (exponential decay rate of first moment estimate) of 0.5, as advised by the authors of DCGAN [2].

Due to the complexity of the networks, we run all of our experiments on two separate GPU enabled devices. We run most of our experiments on a laptop equipped with a NVIDIA GTX 1060 GPU while some training iterations were done using a Amazon Web Service EC2 Deep Learning instance with a NVIDIA K80 GPU. Even though using these machines makes our training much faster, each training iteration takes a considerable amount of time. This results in us having to run our experiments over night and use the days to implement our networks and interpret the results.

3.3 Data Collection

During experimentation we mostly rely on the popular benchmark dataset CIFAR10 [12], however an additional dataset was collected using Flickr's API. Due to varying amounts of quality pictures of objects that would be interesting to base the image generation on, a collection that included mostly different reptiles with a fair amount of arachnid's thrown into the mix was ultimately settled upon, this dataset will be referred to as the Reptiles dataset from here on. In total the data set is made up of approximately 20K color images, all resized to dimensions of 108x108x3 before training was conducted. Examples of images can be seen in figure 3.

Fig. 3: Examples from the Reptiles data set

4 Experiments

The project consists of 8 distinct experiments where each experiment varies from the others either based on what dataset is used or what network settings are used.

4.1 Inception Score Considerations

We present a challenge related to the computation and evaluation of the inception score. Most authors evaluate the inception score on 50K GAN-generated images, as recommend by the authors of the original paper [11]. By running a few preliminary experiments, we quickly realized that on top of the actual training of the network, sampling and computing the inception score are also resource-intensive tasks, and sampling 50K images is simply not possible with the time or resources available for this project.

Now, the number of images considered for evaluating the inception score has an impact on this score, as Table 1 depicts. This is due to the fact that the inception score not only evaluates the content of a given image but also the distribution of categories among the whole set of images resulting from the split. In other words, the score is sensitive to the number of images divided by the number of splits.

Images	Splits	Inception score
256	5	8.13 +- 0.41
512	5	8.04 +- 0.54
1024	5	9.79 +- 0.36

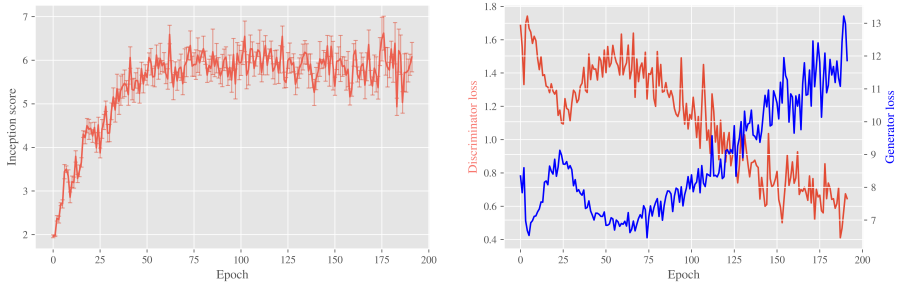
Images	Splits	Inception score
256	10	6.72 +- 0.55
512	10	7.92 +- 0.56
1024	10	8.95 +- 0.44

Table 1: Inception score for various number of samples of the cifar10 dataset.

We choose to stick with 1024 generated images and 5 splits for all of our experiments. With this configuration, we have a target inception score of 9.79. As expected, this is below the claimed inception score of the whole cifar10 dataset, 11.24 [11]. Thus we won't reach state of the art results in terms of inception score, but this isn't an issue since our purpose is to compare various improvements of GAN networks, which isn't affected by this choice. We applied the same calculations to our Reptiles dataset, resulting in a target inception score of $10.24 + -0.93$. Other considerations on the inception score are explained in [17].

4.2 DCGAN

We implement our baseline model in this section. The DCGAN is evaluated on cifar10. We present the evolution of the inception score in Figure 4a and both losses in Figure 4b. These results serve as a reference for all our other experiments. The inception score seems to plateau around 5.8 with considerable deviations. Both the D and G losses are very unstable during training.
It is however important to note that losses can hardly be interpreted when treating with GANs, since the generator and discriminator are in a situation of competition where an improvement on the one leads to a deterioration on the other.



(a) Inception score (b) Losses

Fig. 4: DCGAN - training on CIFAR10 over 190 epochs.

We conduct a visual inspection of the generated images and observe that many of them look alike, if not identical. This is a common phenomenon called

mode collapse, where the generator fails to learn the complete multimodal structure of the target data distribution and rather learns a subset of modes, allowing it to trick the discriminator but not to reproduce the target distribution:

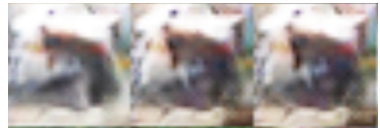
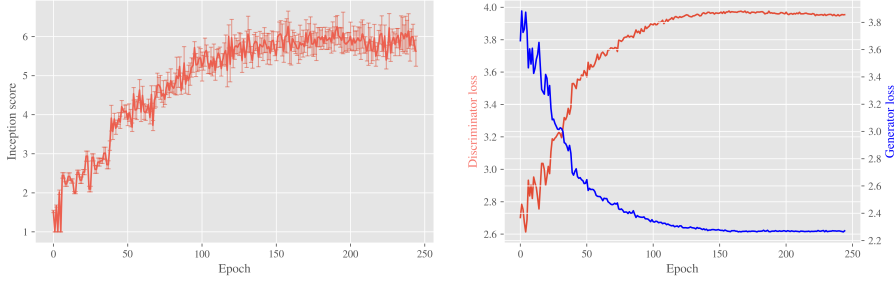


Fig. 5: DCGAN - example of mode collapse consequences.

4.3 SN-DCGAN

We implement spectral normalization and use it on the discriminator of the DCGAN used in the previous section. We present the evolution of the Inception score and losses in Figures 6a and 6b. We observe that both losses are much smoother with spectral normalization. This is the desired result; the original intent of the paper on spectral normalization was to provide a solution to stabilize the training of GANs [10]. By inspecting the generated images we observe a significant reduce in mode collapse and improvement of image quality.



(a) Inception score (b) Losses

Fig. 6: SN-DCGAN - training on CIFAR10 over 250 epochs.

We show example generations on both of our datasets:

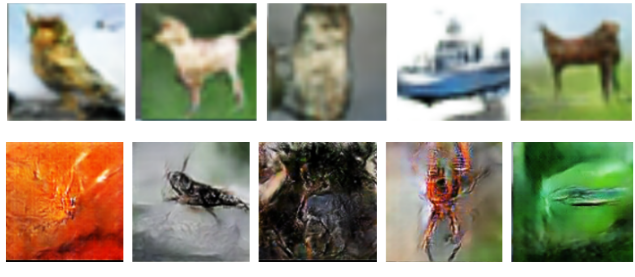
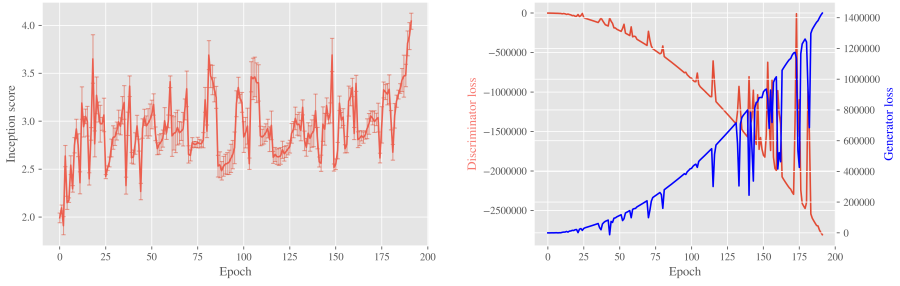


Fig. 7: SN-DCGAN - generated images after training on CIFAR10 (top) and Reptiles (bottom).

4.4 W-DCGAN

We present the evolution of the Inception score and losses in Figures 8a and 8b, respectively. We did not expect to a get good performance using this setup since we did not implement any method enforcing the Lipschitz continuity of the discriminator functions, which is required by WGAN. These results coincide with our expectations.

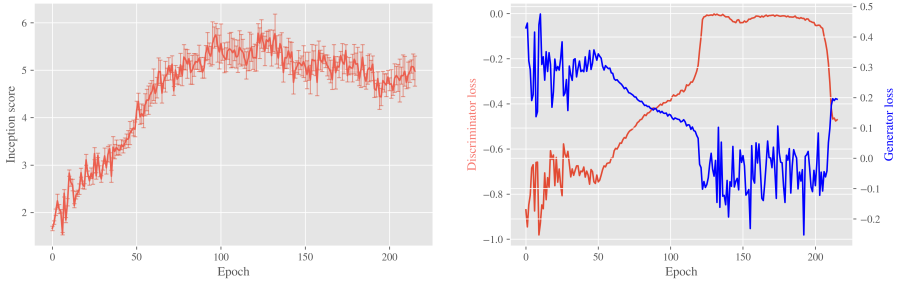


(a) Inception score (b) Losses

Fig. 8: W-DCGAN - training on CIFAR10 over 200 epochs.

4.5 W-WC-DCGAN

We implement weight clipping (WC) on top of the W-DCGAN from the previous section. As stated before, the authors of WGAN state that this method is a far from ideal way to ensure a norm on the gradients. However, we still expect it to outperform the W-DCGAN. We present the evolution of the Inception score and losses in Figures 9a and 9b, respectively.



(a) Inception score (b) Losses

Fig. 9: W-WC-DCGAN - training on CIFAR10 over 200 epochs.

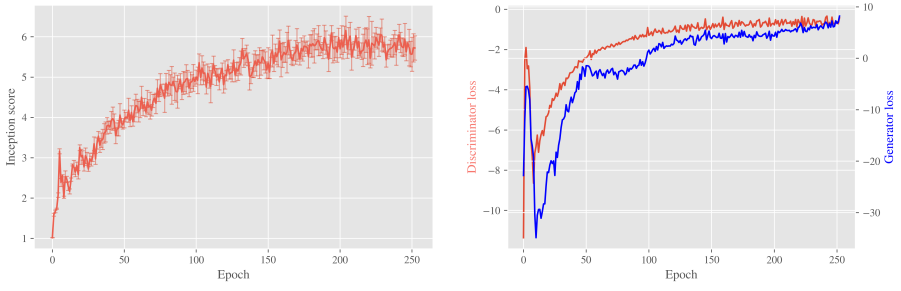
We observe a turbulent climb for the Inception score and an even more chaotic evolution of the losses. Looking at the losses, it seems like the training has two different "regimes". We propose an interpretation, even though we cannot be sure of its accuracy. In all experiments, we initialize weights by sampling from a

normal distribution with mean 0 and standard deviation 0.02. Also, we use the recommended clipping value of 0.01 [8]. This means that for the first iterations, most weights are systematically clipped, and possibly by a lot. It is easy to think how this would lead to unpredictable or unstable behaviour, seemingly experienced in epochs 0 to 50. Then, we think that weights enter a range that suits the requirements of WGAN, leading to a more stable training phase until epoch 120. After this epoch, the generator seems to enter a new chaotic regime, possibly because the discriminator reached a well performing region where it becomes hard to fool. This problem might be solved with thorough hyper-parameter tuning.

4.6 W-SN-DCGAN

We replace weight clipping by spectral normalization. A first experiment with the same hyper-parameters as for all other models has shown very poor performance. By further looking into the literature, we found out that when using an optimizer such as Adam, setting β_1 (the exponential decay rate for the first moment estimate) to 0 was recommended when using the Wasserstein loss [8].

The evolution of the Inception score and losses is reported in Figures 10a and 10b. We observe stable Inception score and loss evolution, compared to the previous methods. We note that the generator loss looks inverted compared to that of the SN-DCGAN, that is simply because of its different formulation. However, the Inception score does not seem to reach significantly higher values than with the SN-DCGAN.



(a) Inception score (b) Losses

Fig. 10: W-SN-DCGAN - training on CIFAR10 over 200 epochs.

We show example generations on both of our datasets:

Fig. 11: W-SN-DCGAN - generated images after training on CIFAR10.

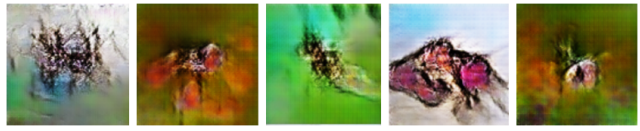


Fig. 12: W-SN-DCGAN - generated images after training on Reptiles. We could not train for a sufficient time; these images show usual signs of the early stages of GAN training, e.g. repetitive patterns within the image and unclear object.

5 Conclusions

Figure 13 compares the Inception score for all tested network settings. Even though the Inception score of the basic DCGAN is not improved using the evaluated methods the overall quality of the generated images is clearly higher and the presence of mode collapse is significantly reduced or even non-existent for GAN's where spectral normalization, Wasserstein loss or both were added.

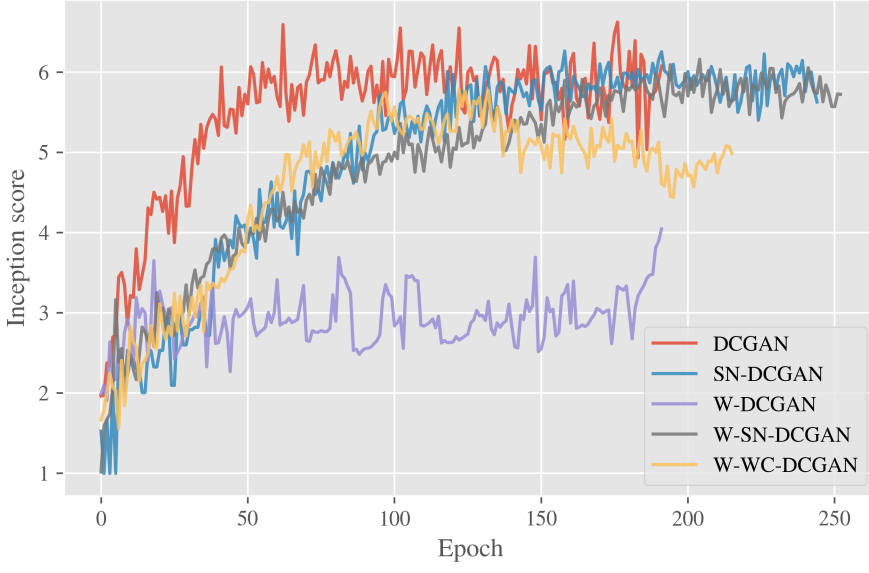


Fig. 13: All evaluated models - Inception score, training on CIFAR10

In addition to these improvements, both the D loss and G loss were much smoother during training. Having smoother losses allows for much better monitoring of the networks behaviour during the training process, leading to a much more intuitive hyper parameter tuning and debugging process.

Using our two datasets allowed us to better compare effect of each addition on the DCGAN. In fact, the Reptiles dataset consisting of images with much

higher resolution, it required more time to train GANs using it but also showed more interesting and diversified image generations.

5.1 Future Work

In the future, we would try and further explore the parameter space. The training of GANs is always a lengthy process with many influencing factors. With more time and resources, we would run parameter searches in a more systematic way and hopefully find better performing configurations. More specifically, we would start lower learning rates (ours might have been slightly too high; experiments show harsh loss evolutions) or different optimizers (WGAN literature advices to use RMSProp [8]).

Another contribution to our work would be to implement gradient penalty and run an experiment with Wasserstein loss as well. We expect that the performance of this configuration should fall in between that of our W-WC-DCGAN (weight clipping) and W-SN-DCGAN (spectral normalization).

Finally, we would apply the same methodology to the LSGAN framework [13] and see if spectral normalization can also help making training more stable.

References

1. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. (2014) 2672–2680
2. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. CoRR **abs/1511.06434** (2015)
3. Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al.: Photo-realistic single image super-resolution using a generative adversarial network. arXiv preprint (2016)
4. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. arXiv preprint (2017)
5. Kaneko, T., Kameoka, H., Hojo, N., Ijima, Y., Hiramatsu, K., Kashino, K.: Generative adversarial network-based postfilter for statistical parametric speech synthesis. In: Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on, IEEE (2017) 4910–4914
6. Pascual, S., Bonafonte, A., Serra, J.: Segan: Speech enhancement generative adversarial network. arXiv preprint arXiv:1703.09452 (2017)
7. Guimaraes, G.L., Sanchez-Lengeling, B., Farias, P.L.C., Aspuru-Guzik, A.: Objective-reinforced generative adversarial networks (organ) for sequence generation models. arXiv preprint arXiv:1705.10843 (2017)
8. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein gan. arXiv preprint https://arxiv.org/abs/1801.01973 (2017)
9. Uehara, M., Sato, I., Suzuki, M., Nakayama, K., Matsuo, Y.: Generative adversarial nets from a density ratio estimation perspective. arXiv preprint arXiv:1610.02920 (2016)
10. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks. arXiv preprint arXiv:1802.05957 (2018)

11. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training gans. In: Advances in Neural Information Processing Systems. (2016) 2234–2242
12. Krizhevsky, A., Nair, V., Hinton, G.: Cifar-10 (canadian institute for advanced research)
13. Mao, X., Li, Q., Xie, H., Lau, R.Y., Wang, Z., Smolley, S.P.: Least squares generative adversarial networks. In: 2017 IEEE International Conference on Computer Vision (ICCV), IEEE (2017) 2813–2821
14. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. CoRR **abs/1512.00567** (2015)
15. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: AISTATS. (2011)
16. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR **abs/1412.6980** (2014)
17. Barratt, S., Sharma, R.: A note on the inception score. arXiv preprint arXiv:1801.01973 (2018)