Evaluation of Spectral Normalization for GANs Using Inception Score

Eysteinn Gunnlaugsson, Egill Vignisson, Charles Hamesse

Group 16

Abstract. **Todo:** write after W-SN-DCGAN We experiment various Deep Convolutional Generative Adversarial Networks (DCGANs) for image generation. We start with a vanilla DCGAN and gradually add features that are expected to improve learning in terms of speed, stability and quality of generation. We evaluate our models using the inception score and discuss its relevance on cifar10 and a dataset that we collected ourselves consisting of reptile images. Finally, we discuss our findings and challenges and draw conclusions on how the features we implemented help the training of GANs.

Keywords: Generative models, generative adversarial networks, image generation

1 Introduction

Todo: write after W-SN-DCGAN Motivate the problem you are trying to solve, attempt to make an intuitive description of the problem and also formally define the problem. (1-2 pages including title, authors and abstract)

The purpose of this project is to investigate the performance of the different types of Generative Adversarial Networks (GANs) [1] for image generation as well as possible improvement options. The project was originally defined based on varying levels of priority where everything assigned priority 1 was promised to be completed.

- Implement Deep Convolutional Generative Adversial Network with original loss [2] (priority 1)
- Implement the inception score metric [3] (priority 1)
- Implement Spectral Normalization (priority 1)
- Evaluate all our GANs on our reptile dataset (priority 1)
- Implement other losses (LSGAN, WGAN) [4] (priority 2)
- Evaluate GANs on CIFAR-100 (priority 2)
- Implement mini-batch discrimination and or other improvements [3]. (priority 3)

In order to evaluate the performance of these GANs, the evaluation metric known as the inception score, as described in [3], will be implemented. Furthermore a data set consisting of roughly 30K animal pictures (mostly reptiles), was

fetched from the Flickr API while other well known options such as a subset of CIFAR-100 or ImageNet were thought of as possible replacements in the case of unsatisfactory results.

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.

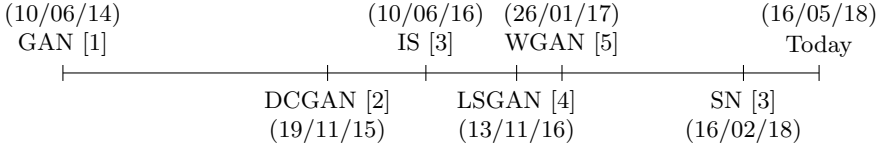


Fig. 1: Major contributions in the field of Generative Adversarial Networks.

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:

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

[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.

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 [3]. 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, which making it difficult to compare the performance of different models. In the context of image generation, an intuitive 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 [6] 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:

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

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

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

$p(y|\boldsymbol{x})$ 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:

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

2.4 WGAN

Todo: write

Todo: cite authors of WGAN saying "Weight clipping is a clearly terrible way to enforce a Lipschitz constraint"

Recommended clip limit is 0.01

2.5 Spectral Normalization

Spectral normalization [7] 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 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 **Todo:** cite), 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 $\|f\|_{\text{Lip}}$:

$$\|f\|_{\text{Lip}} \leq \prod_{l=1}^{L+1} \|a_l\|_{\text{Lip}} \|g_l\|_{\text{Lip}} = \prod_{l=1}^{L+1} \sigma(W^l). \tag{5}$$

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

$$\bar{W}_{l\text{SN}}(W_l) := W_l / \sigma(W_l), \tag{6}$$

enforcing the upper bound of 1 for $\|f\|_{\text{Lip}}$ using the inequation 5.

[2] As a matter of fact, this has been posted on the peer-reviewing platform `https://openreview.net/forum?id=B1QRgziT-`:

> *"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 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.

3.1 Network Architecture

Generator architecture details. The generator's 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(Eq.(7)) while the output layer uses the tanh function(Eq.(8)). 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"*(p. 3).

Discriminator architecture details . The discriminator's architecture is basicly follows a very similar structure. .

$$x = max(0, x) \tag{7}$$

$$\tanh x = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{8}$$

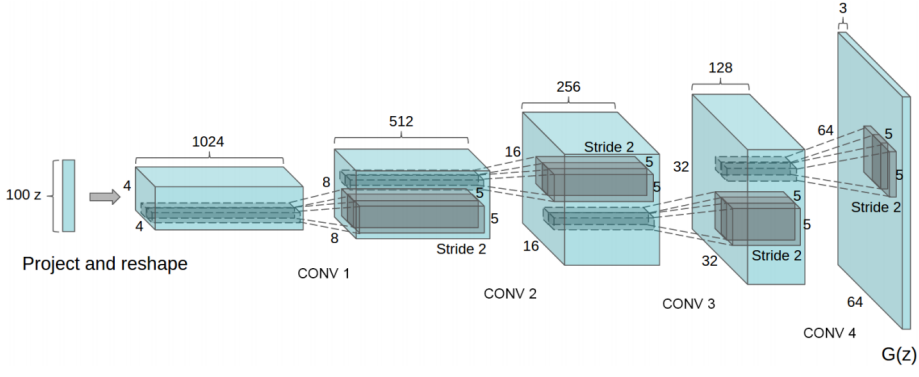


Fig. 2: Generator Network architecture. Cited from [2]

3.2 Network Training

Due to the complexity of the networks we did all of our experiments on two separate GPU enabled devices. We ran most of our experiments on a laptop that has 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 utilizing these machines made our training much faster, each training iteration took a considerable amount of time. This resulted 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 relied on the popular benchmark dataset CIFAR10, however an additional data set 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 therefore this data set will be referred to as the Reptiles form here on. In total the data set is made up of approximately 20K color images all re-sized 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

3.4 Frame of Reference

In order to measure the performance of each implementation a frame of reference had to be established. As the golden standard, that every implementation would be compared to, the inception score of the actual data sets were calculated, the resulting scores can be seen in table 1

Table 1: Inception scores for the data sets

CIFAR10	Reptiles
9.792953 +- 0.36040735	10.2415905 +- 0.92625165

4 Experiments

In this section, you should present the results you achieved with various experiments. The results can be presented in tables, plots, etc.

The purpose of the project is to analyze the nature and effectiveness of different GAN architectures as well as different improvement options. The different models but in order to do so a frame of reference was needed

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 recommended by the authors of the original paper [3]. By running a few preliminary experiments, we quickly realize 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 shows Table 2. 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 amongst 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

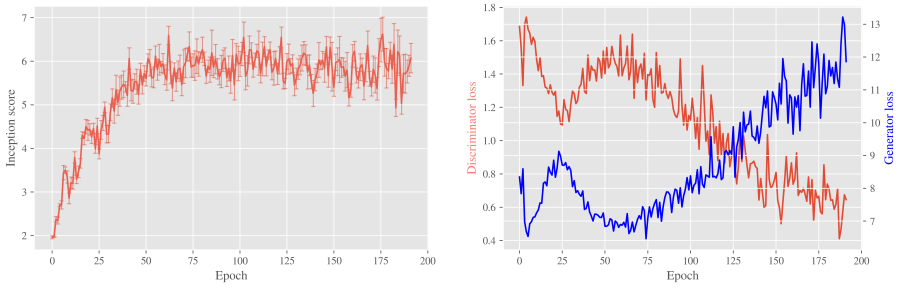
Images	Splits	Inception score
256	10	6.72 +- 0.55
512	10	7.92 +- 0.56
1024	10	8.95 +- 0.44

Table 2: 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 [3]. 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. Other considerations on the inception score are explained in [8].

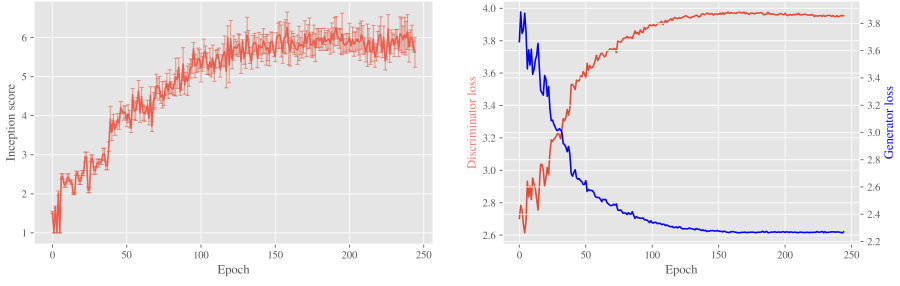
4.2 DCGAN

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



(a) Inception score (b) Losses

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



(a) Inception score (b) Losses

Fig. 5: SN-DCGAN - training on CIFAR10 over 250 epochs.

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.

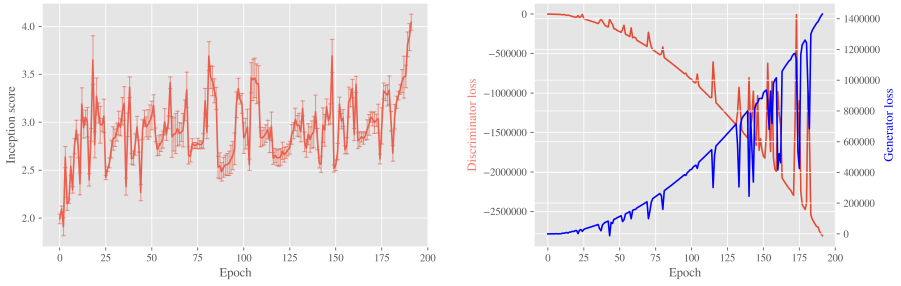
4.3 SN-DCGAN

We implement the spectral normalization layer and use on the discriminator of the DCGAN used in the previous section. We present the evolution of the inception score and losses in Figures 5a and 5b, respectively.

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 [7].

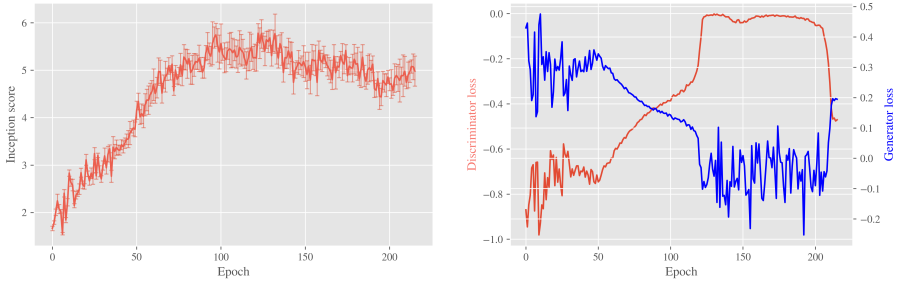
4.4 W-DCGAN

We present the evolution of the inception score and losses in Figures 6a and 6b, respectively.



(a) Inception score (b) Losses

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



(a) Inception score (b) Losses

Fig. 7: W-WC-DCGAN - training on CIFAR10 over 250 epochs.

4.5 W-WC-DCGAN

We present the evolution of the inception score and losses in Figures 7a and 7b, respectively.

4.6 W-SN-DCGAN

We present the evolution of the inception score and losses in Figures 8 and 9, respectively.

Fig. 8: W-SN-DCGAN - Inception score, training on cifar10 over 250 epochs

4.7 Performance Comparison

Fig. 9: W-SN-DCGAN - Losses training on cifar10 over 250 epochs.

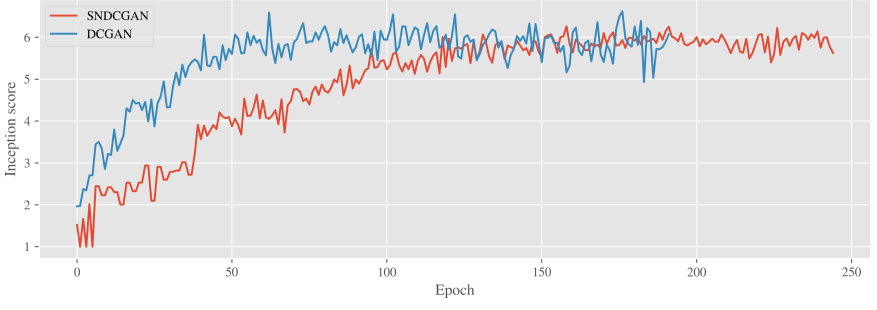


Fig. 10: All evaluated models - Inception score, training on cifar10

5 Conclusions

Explain what conclusions you can draw from these set of experiments? The set of experiments and results reported here should justify some of the design choices described in the previous sections. (3-6 pages)

5.1 Future Work

- LSGAN - Gradient penalty - Parameter search

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. 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
4. 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
5. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein gan. arXiv preprint https://arxiv.org/abs/1801.01973 (2017)
6. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. CoRR **abs/1512.00567** (2015)
7. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks. arXiv preprint arXiv:1802.05957 (2018)
8. Barratt, S., Sharma, R.: A note on the inception score. arXiv preprint arXiv:1801.01973 (2018)