WASSERSTEIN GENERATIVE ADVERSARIAL NETWORKS

Jens Christian Bang Gribsvad (s174477), Matias Bundgaard-Nielsen (s173981)

DTU
Compute
Richard Petersens Plads, Bygning 324,2800 Kgs. Lyngby

ABSTRACT

In this paper, we perform a detailed comparison between Generative Adversarial Networks (GAN) and one of their variations: the Wasserstein GAN. We compare their performance and training convergence on MNIST, CIFAR10, and CelebA using the same architecture and only changing the loss function. We consider three different ways of enforcing the Lipschitz continuity of WGANs: weight clipping, spectral normalization, and gradient penalty. We find no significant difference in the performance of GAN and WGAN. This is likely due to the architectures being optimized for the GAN model and then later adopted for the WGAN. We find that spectral normalization has significantly worse performance than the other methods of enforcing Lipschitz continuity, which might be due to the use of layer normalization instead of batch normalization.

For details on network architectures and reproducing all results in the paper, we refer to our public GitHub repository: `https://github.com/mabuni1998/DeepLearningGANProject`

1. INTRODUCTION

Since the inception of the internet, the amount of easily accessible data has exploded. This, together with the increasing power of computers, has led to impressive results in supervised learning tasks such as image [1] and speech [2] recognition. The vast majority of data available is, however, unlabeled, and since labeling is often a tedious and expensive task, unsupervised learning has a significant interest. One area of unsupervised learning is Generative Adversarial Networks (GAN) [3], where a generator tries to fool a discriminator with its generated images. This leads to the generator learning the training set distribution without any supervision. GANs have many exciting use cases, the most apparent being generation of more training data [4], but success has also been found in super-resolution [5] and inpainting [6].

GANs are, however, infamous for being hard to train. More specifically, they can be unstable and never reach convergence [7, 8]. One proposed solution is to use the Wasserstein distance in a so-called Wasserstein GAN (WGAN), which theoretically has better convergence. In a WGAN, the discriminator, however, needs to be Lipschitz continuous and this constraint can be enforced in different ways, with some being weight clipping [8], spectral normalization [9], and gradient penalty [10].

In this paper, we consider a GAN and WGAN and compare their performance and training convergence on three datasets: MNIST, CIFAR10, and CelebA. Furthermore, we study three different ways of enforcing the Lipschitz continuity in WGANs (weight clipping, spectral normalization and gradient penalty) and how this impacts training. For each dataset, the network architecture is identical across the GAN and WGANs, so only the loss function or the method enforcing the Lipschitz continuity is changed. We find that there is little difference in performance and that convergence happens at roughly the same rate. However, this does not mean that WGANs do not live up to their expectations as easier to train since we, in all cases, optimized the architecture for the GAN and then changed the loss function and turned it into a WGAN.

This paper is organized as follows: In sec. 2, we introduce the GAN and WGAN structures and the relevant theory behind their loss functions. In sec. 3, we show the results of our training across 10 runs on each dataset for each network. In sec. 4, we discuss the results and compare the different networks, followed by a conclusion in sec. 5.

2. MODEL/ THEORY

The goal of a GAN is to produce fake images that resemble true images. The structure of a GAN consists of two neural networks; a generator and a discriminator. A sketch of the structure can be seen in fig. 1. The discriminator aims to distinguish between real and fake images, whereas the generator aims to create real-looking images from random noise and fool the discriminator into labeling these fake images as true images. Essentially,

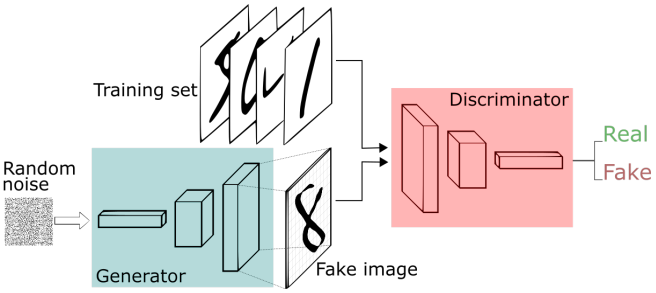

Fig. 1: Illustration of the structure of a GAN. Random noise is put into the generator, which turns the noise into a fake image. The fake image is then together with a real training data set fed into a discriminator which tries to distinquish the inputs as real or fake. Taken from ref. [11].

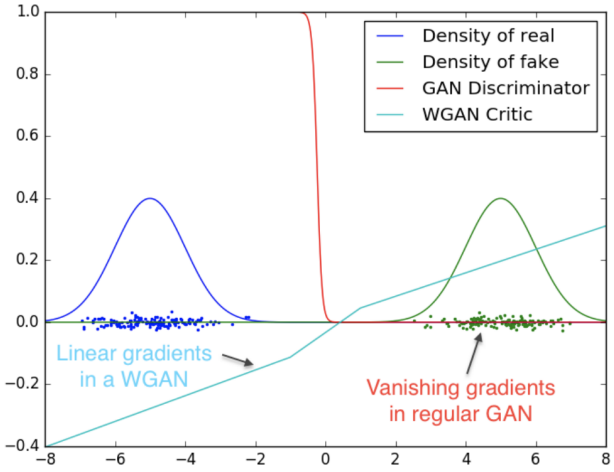

Fig. 2: Graph showing the difference between the GAN and WGAN loss function when considering a real and fake distribution. It is clear that the GAN has a vanishing gradient, while the WGAN has a more linear gradient that is easier to train on. Picture was taken from ref. [8].

the generator tries to learn the distribution of the true images and minimize the distance between the true-image distribution and the fake-image distribution. Contrary, the loss function of a discriminator is centered around separating the two distributions from each other. There exist many ways of measuring the difference between two distributions, but we will consider two; the Jensen-Shannon divergence used in GAN and the earth-mover distance used in WGAN. [3]

To introduce the aforementioned loss functions, we define the two distributions $\mathbb{P}_r$ and $\mathbb{P}_g$, which later will represent the distribution of the real data and the distribution of the generated data. The Jensen-Shannon divergence can be written as [3]:

$$\mathrm{JS}(\mathbb{P}_r, \mathbb{P}_g) = \mathrm{KL}(\mathbb{P}_r||\mathbb{P}_m) + \mathrm{KL}(\mathbb{P}_g||\mathbb{P}_m) \tag{1}$$

where KL is the Kullback-Liebler divergence [3] and $\mathbb{P}_m = (\mathbb{P}_r, \mathbb{P}_g)/2$ is the mixture of the two distributions. In the GAN, the generator tries to minimize the JS-divergence, whereas the discriminator tries to maximize it. In reality, the loss function which is used for training is, for practical reasons, defined as the Binary Cross Entropy loss:

$$\mathcal{L}_{GAN} = \mathbb{E}[\log(D(X))] + \mathbb{E}[\log(1 - D(G(Z))] \tag{2}$$

where $\mathbb{E}$ is the mean, X is the test distribution, and Z is the random noise input of the generator. One can show that minimizing (maximizing) this loss function for the generator (discriminator) is equivalent to minimizing (maximizing) the Jensen-Shannon divergence [3]. Since the labeling is true or false, the last output layer of the discriminator has a sigmoid activation function.

In the WGAN, one instead seeks to minimize (maximize) the earth-mover distance or Wasserstrin-1. Using the Kantorovich-Rubinstein duality, this can be written as [8]:

$$\mathcal{L}_{WGAN} = \sup_{||f||_L \leq 1} \mathbb{E}[f(X)] - \mathbb{E}[f(G(Z))] \tag{3}$$

where $||f||_L$ is the Lipschitz norm, and f(X) is now the Wasserstein discriminator or, more appropriately, critic since the output is no longer true or false but rather a score. A high positive (negative) score thus means that the critic thinks it is a real (fake) image. Furthermore, there is the restriction that the critic obeys the Lipschitz norm or is Lipschitz continuous. The primary motivation behind using the Wasserstein loss is that it combats vanishing gradients, which can be a problem in the standard GAN loss function. This is also illustrated in fig. 2, where a fake distribution is easily distinguishable from a real distribution leading to a vanishing gradient in the GAN discriminator.

A challenge when implementing a WGAN is to enforce Lipschitz continuity. We here consider three different methods; Spectral Normalization, Weight Clipping, and Gradient Penalty. Spectral normalization enforces Lipschitz continuity by re-scaling the weights in the discriminator, so all linear layers have singular values smaller than 1 [9]. Weight clipping enforces Lipschitz continuity by scaling weights to be within a pre-defined interval. This pre-defined interval is a hyperparameter that can be tuned. We found [-0.02,0.02] to work best which is similar to the proposed interval of [-0.01,0.01] in [8]. Gradient penalty enforces Lipschitz continuity by forcing the gradient of the discriminator loss function to have a unit norm. This is done by adding a new term to

the loss function [10]:

$$\mathcal{L}_{GP} = \lambda \mathbb{E}(\|\nabla f(Y)\|_2 - 1)^2]$$

Here Y is the distribution obtained by uniformly sampling along a straight line between $\mathbb{P}_r$ and $\mathbb{P}_g$ whereas λ is the penalty coefficient which is another hyperparameter that can be tuned. We use $\lambda = 10$, which fits many architectures and data sets according to [10].

The choice of loss function significantly impacts the training of a GAN. In a standard GAN, a common problem is, as mentioned, vanishing gradients. If the discriminator is too strong and correctly classifies all images, the sigmoid activation function leads to vanishing gradients, and the generator never learns the distribution. Moreover, mode collapse is another issue that occurs in GANs where the generator produces images that are not as diverse as the true images, i.e., only producing ones and zeros when trained on the whole MNIST dataset. These problems are proposed to be solved by WGANs [8]. The more gradual nature of the loss function, as opposed to either true or false in a standard GAN, leaves a more linear gradient, and the generator can more easily find a minimum and is thus easier to train. We again refer to fig. 2. The claim of no mode collapse is more of an empirical observation than theoretical, but the main point of WGANs being easier to train still stands.

Finally, it is relevant to have a metric for evaluating the quality of the images produced by the generator. The loss function itself is not a good measure since the generator and discriminator are trained simultaneously and have opposite goals. A very low loss does, therefore, not necessarily mean that the generator successfully generated realistic pictures. Instead, the discriminator could also just be doing an extremely poor job of judging the images. A common practice is instead to use the Fréchet Inception Distance (FID) [12] since it captures the similarity images by feeding them into a third network: the Inception V3 Model. Another measure is the Inception Score, but we will not consider it since it is inferior to the FID [12]. The FID is computed by taking the true and generated images and feeding them into an Inception V3 model, which has been trained on the ImageNet data. The last layer of the model is removed, so the output of the model is a feature vector. The feature vectors are then used to compute:

$$\text{FID} = \|\mu_1 - \mu_2\|^2 + \text{Tr}\left(C_1 + C_1 - 2\sqrt{C_1 C_2}\right) \qquad (4)$$

where Tr is the trace, μ is the mean, and C is the covariance matrix of the feature vectors. A low FID indicates similar distributions and, thus a higher quality of the generated images.

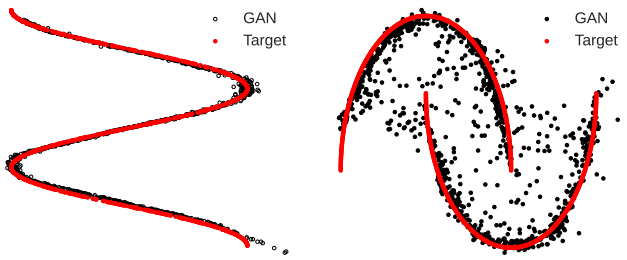

Fig. 3: GAN learning s-curve (left) and half moon (right) distributions.

3. RESULTS

The first experiment we considered was to train a GAN to learn two simple 2D distributions. The computational power required to train on 2D distributions is low, and the architecture of the discriminator and generator can be quite simple. This allows fast training, and thus it's easier to catch errors in the implementation. Figure 3 displays the results of a GAN learning an s-curve and half-moon distribution. The GAN learns both distributions quite well, however, the generated half-moon data is not as similar to the target distribution as the s-curve distribution. As stated in [13], GANs tend to have difficulty learning discontinuous distributions, which explains the difference in the results between the two distributions. With a simple architecture working on the distributions, we were then able to increase the complexity of the architecture and move on to images.

We considered 3 different data sets; MNIST, CIFAR10, and CelebA. To decrease the computational time, we have chosen to rescale the CelabA images from 178×218 pixels to 64x64 pixels. The performance of the 4 models across the 3 data sets can be seen in fig 4. The FID is quite similar across each data set, with the exception of WGAN-SN, which tends to have a larger FID on all three data sets. Moreover, it is evident that the models produce the best results on the MNIST data set, whereas they struggle a bit more on CIFAR10 and CelebA where the FIDs are larger.

The best results we obtained on the MNIST data set were with the GAN model. It obtained a FID of 12.325. In figure 5, a sample of generated images can be seen, and it is evident that the GAN has successfully learned to produce numbers, although occasionally, some numbers are illegible.

We obtained the best results for CIFAR10 and CelebA with WGAN-GP, which obtained FIDs of 72.416 and 50.253, respectively, although the average performance was better with the standard GAN. The generated images by WGAN-GP for the CIFAR10 data set look some-

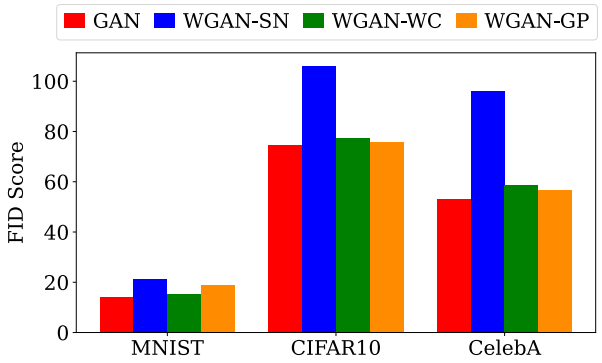

Fig. 4: Model performance across MNIST, CIFAR10 and CelebA data sets. The FID score shown is an average over 10 experiments

what realistic with clear structures in each image, but the object in the images can be difficult to determine, as seen in figure 5. Based on visual inspection, we found that birds and cars are the best-represented classes, but overall the images are far from realistic. For CelebA, the FID is lower, and the quality of the images is better, as seen in fig. 5. The generated images display human faces, but the quality is not perfect. Some faces have black patches as eyes, whereas some faces have hair that is twisted and unnaturally edgy. However, it is evident that the WGAN-CP managed to generate human faces similar to the CelebA data set. Although CIFAR10 is a simpler dataset than CelebA, the wide variety of pictures in CIFAR10 (a dog is much different looking than a car) seems to make it harder for the GAN to learn the distribution than the more homogeneous celebrity faces that are all quite similar looking.

Considering now the convergence during the training of the models. We have displayed the convergence for each model on each data set across 10 separate experiments in figure 6. For the MNIST data set, it can be seen the models converge equally fast around the first few epochs. However, for the CIFAR10 and CelebaA data sets, the WGAN-SN converges slower and struggles to obtain the same low FID as the other models. Moreover, it can be observed that WGAN-GP converges a bit slower in the CelebA data set than in the other models.

4. DISCUSSION

As stated in section 1 GANs are known to be hard to train. Thus, we started this project by trying to find a GAN that could converge for the 2D distributions and later for the image data sets. Inspired by [14] and manual tuning, we found an architecture and hyperparameters that could make the GAN converge. Then we created

Fig. 5: Generated images for MNIST, CIFAR10 and CelebA, using the GAN with FID = 12.325, WGAN Gradient Penalty with FID = 72.416, and FID = 50.253 respectively.

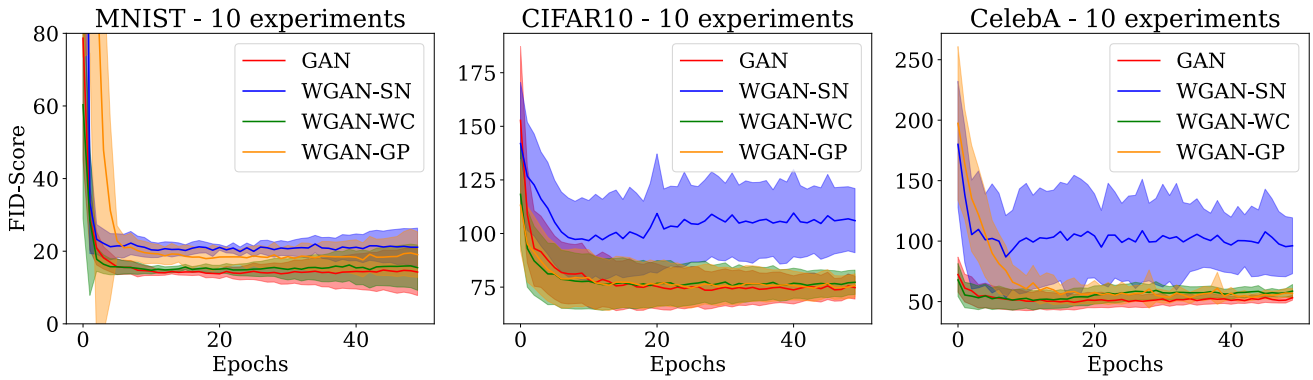

Fig. 6: Confidence intervals based on averages of 10 experiments for the four models across the 3 data sets.

WGANs with the exact same architecture and hyperparameters and saw that the WGANs also converged. However, it can be argued that this approach favors the GAN since we did not search for a more optimal architecture and hyperparameters for the WGANs, which quite possibly could give us better results. In general, a comprehensive hyperparameter search for both the GAN and WGAN would be desired to obtain the most accurate comparison. We did also find that for some architectures, we could make the WGAN converge, while the GAN was not able to, which somewhat confirms the assumption that WGANs are easier to train than GANs. These results are not included here.

Another aspect of the discussion is the performance of the WGAN-SN, which was found to be much worse in the CIFAR10 and CelebA data sets. We believe the use of *layernorm* instead of *batchnorm* is the reason behind this. From our experience, *batchnorm* does not work well together with spectral normalization as we were not able to see any convergence at all using such a configuration. We are therefore left with *layernorm*, which possibly hurts performance or at least requires a different architecture.

As mentioned in section 2, mode collapse can be an issue for GANs. In further work, one could do a qualitative analysis of the output for both a GAN and WGAN and investigate mode collapse. Empirically, it has been found in [8] that WGANs do not exhibit mode collapse, and it would therefore be interesting to see if one could observe the same. To investigate this, we would suggest the following approach: find pre-trained networks that can classify classes within MNIST and CIFAR10 to determine the class of a generated image. Then compare the distribution of classified classes for a number of generated images to see if some classes are underrepresented (i.e., mode collapse). Do this for both a GAN and WGAN and compare the results.

5. CONCLUSION

In this project, we have studied the dynamics of GANs, and in particular, the min-max game of the loss functions between the generator and discriminator. Furthermore, we have investigated how a WGAN works and how different models can enforce Lipschitz continuity, which is a condition for the Wasserstein loss function. We found that a WGAN was easier to train and provided similar FIDs as a GAN. Finally, we can conclude that we successfully trained a GAN, WGAN-WC, WGAN-GP, and WGAN-SN to generate realistic-looking images and distributions resembling true data.

6. REFERENCES

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, Eds. 2012, vol. 25, Curran Associates, Inc.

[2] Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, and Brian Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling,

C. Cortes, N. Lawrence, and K.Q. Weinberger, Eds. 2014, vol. 27, Curran Associates, Inc.

[4] Maayan Frid-Adar, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan, "Synthetic data augmentation using gan for improved liver lesion classification," in *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*. IEEE, 2018, pp. 289–293.

[5] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi, "Photo-realistic single image super-resolution using a generative adversarial network," 07 2017, pp. 105–114.

[6] Dongmin Cha and Daijin Kim, "Dam-gan: Image inpainting using dynamic attention map based on fake texture detection," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 4883–4887.

[7] Martin Arjovsky and Léon Bottou, "Towards principled methods for training generative adversarial networks," *stat*, vol. 1050, 01 2017.

[8] Martin Arjovsky, Soumith Chintala, and Léon Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*. PMLR, 2017, pp. 214–223.

[9] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida, "Spectral normalization for generative adversarial networks," *arXiv preprint arXiv:1802.05957*, 2018.

[10] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville, "Improved training of wasserstein gans," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2017, NIPS'17, p. 5769–5779, Curran Associates Inc.

[11] "A short introduction to generative adversarial networks," `https://sthalles.github.io/intro-to-gans/`, Accessed: 2022-14-12.

[12] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in neural information processing systems*, vol. 30, 2017.

[13] "Deep learning course, 11. generative adverserial models," `https://fleuret.org/dlc/`, Accessed: 2022-09-12.

[14] Alec Radford, Luke Metz, and Soumith Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.