NEVER TRUST A GAN: A COMPARATIVE STUDY OF GAN, WGAN AND WGAN WITH SPECTRAL NORMALIZATION

Liu, Haotian
s212645@student.dtu.dk

Nistor, Mihaela-Elena
s183190@student.dtu.dk

Taghikhani, Milad
s212518@student.dtu.dk

Ølgaard, Kristine
s201952@student.dtu.dk

ABSTRACT

This paper presents a comparative study between Generative Adversarial Networks(GANs), WGANs, and WGANs with spectral normalization. The three models were trained on MNIST, FashionMNIST, and CIFAR-10 datasets, and their performance was evaluated using the Frechet Inception Distance (FID) score. The results show that WGAN performs better than DCGAN and SN-WGAN on the CIFAR and MNIST datasets, and SN-WGAN delivers better results on the FashionMNIST dataset.

1. INTRODUCTION

Generative Adversarial Networks (GANs) provide powerful networks for learning data distributions of image, video, and voice generation. GANs have applications in many different domains such as art generation, simulation of aging, or for improving the performance of other machine learning models by generating synthetic training data.
A comparative study between GAN and WGAN is useful in highlighting the key differences in the training process as well as the quality of the generated data. WGAN brings interesting improvements to both which makes it a suitable candidate model for many different tasks, while GAN is known for producing high-quality realistic samples, which means that it is preferred over WGAN for certain given applications.

2. LITERARY REVIEW

Generative Adversarial Networks (GAN) have been introduced in 2014 by Ian J. Goodfellow [1] offering a fresh perspective on generative models via an adversarial framework, where two models, a generator and a discriminator are pitted against each other with the goal of estimating distributions of datasets encountered in artificial intelligence application such as visual and audio data. The sample results obtained on MNIST, Toronto Face Database (TFD), and CIFAR-10 prove GAN to be competitive with other generative models in the literature. Adversarial models present computational advantages as well as disadvantages that show that without good synchronization between the discriminator and generator during training, the discriminator stops learning, and the generator presents the discriminator with a small set of output examples also known as mode collapse. For generative models, the process of estimating a data distribution involves minimizing the distance between the generated and real data distribution. Wasserstein-GAN [2] is a new take on generative adversarial networks that propose trading the Jensen-Shannon divergence for an approximation of the Earth Mover (EM) distance, which represents a more sensible cost function that gives good results even when learning distributions with low dimensional manifolds. The resulting model shows improved stability and no evidence of mode collapse as experiments were carried out on the LSUN-bedroom dataset. This comparative study aims to contribute to existing work [3], [4] by providing a comparison of GAN with WGAN supported by experimental results obtained on three well-known datasets MNIST, FashionMNIST, and CIFAR-10.

3. THEORY

3.1. Generative Adversarial Network (GAN)

The generative adversarial network framework consists of a Generator (G) and a discriminator (D) that play a minimax game with value function V(G, D):

$$\min_G \max_D V(D,G) = E_{x\sim p_{\text{data}}(x)}[\log D(x)] + E_{z\sim p_z(z)}[1-\log D(G(z))] \quad (1)$$

D is trained to maximize the probability of assigning the correct label to the training examples and generated samples from G. At the same time G is trained to minimize the quantity $[1-\log D(G(z))]$, which intuitively means fooling the discriminator that the generated example comes from the real distribution. It is proved that the convergence of the generator $C(G)$ is achieved if $V(D,G) = -log4$, which corresponds to $D(x) = \frac{1}{2}$ and $D(G(z)) = \frac{1}{2}$. Assigning a $\frac{1}{2}$ probability to both the real and generated examples shows that the

discriminator is equally uncertain about which data is real and which is generated. This implies that the two distributions are the same, and that is when the divergence measure (Jensen-Shannon divergence) tends to zero.

3.2. Wasserstein Generative Adversarial Network (WGAN)

The choice of divergence function has a high influence on the convergence of the probability distribution. Hence, the Earth-Mover (EM) distance is a continuous cost function where $\Pi(P_r, P_g)$ represents the set of all joint distributions $\gamma(x, y)$ with marginals P_r and P_g. Intuitively $\gamma(x, y)$ represents the amount of mass to be transported from x to y to transform distribution P_r into distribution P_g. EM represents the cost of the optimal transport plan.

$$W(P_r, P_g) = \left[\inf_{\gamma \in \Pi(P_r, P_g)} E_{(x,y)\sim\gamma} \left[\|x - y\| \right] \right] \quad (2)$$

As the infimum in 2 is intractable, an approximation of the EM distance is used in practice, that satisfies the constraint of 1-Lipschitz continuity.

4. IMPLEMENTATION

4.1. Architecture

A GAN consists of two neural networks: a generator and a discriminator. The generator uses the same architecture for DCGAN, Wasserstein GAN, and Spectral Normalization WGAN. The difference between these three models is mainly reflected in the discriminator.

The generator is constructed based on the generator of DCGAN. Transposed convolutional layers are used in DCGAN as upsampling layers. The generator started with a transposed convolutional layer with 4×4 kernel size, 1 stride, and 0 paddings. The input noise will change from a 1×1 tensor with 100 channels to a 4×4 tensor with 1024 channels after passing through the first convolutional layer. This is followed by a batch normalization layer and a rectifier(ReLu) layer. After the ReLu layer, three similar transposed convolutional neural networks are constructed with a transposed convolutional layer with 4×4 kernel size, 2 strides, and 1 padding, a batch normalization layer, and a ReLu layer respectively. The tensor doubles in size after each transposed convolutional layer. The final transposed convolutional network is constructed with a transposed convolutional layer with 3×3 kernel size, 1 stride, 1 padding, and a hyperbolic tangent(Tanh) layer. The output tensor is a tensor with a size of 32×32 and a channel number of 3. The architecture of the generator is shown in Fig.1.

Discriminators of DCGAN, WGAN, and spectral normalization WGAN are different. The downsampling of these three discriminators all uses convolutional neural networks. The convolutional neural network of DCGAN's discriminator

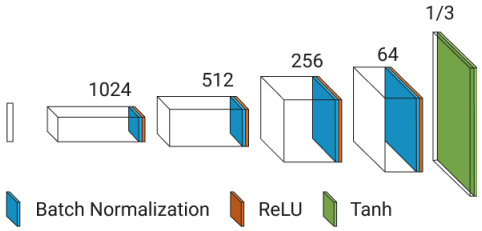
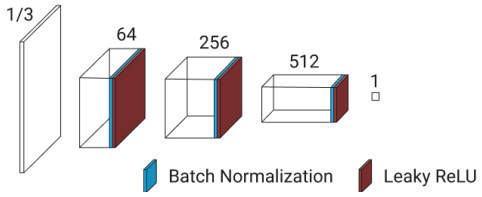
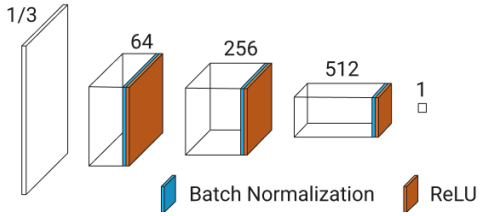


Fig. 1. Generator Architecture

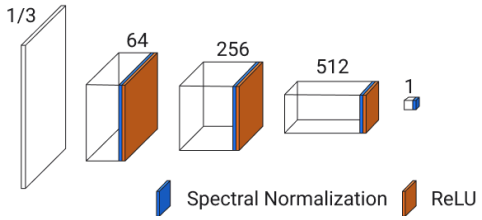
consists of a convolutional layer with a convolution kernel of 4×4, 1 stride and 1 padding, a batch normalization layer, and a ReLu layer. The last two layers of DCGAN's discriminator are convolution kernel 4×4, 1 stride, 0 paddings convolution layer, and Sigmoid activation function layer. Unlike the discriminator of DCGAN, the discriminator of WGAN only removes the last layer, that is, the Sigmoid activation function layer. Spectral normalization WGAN's discriminator replaces all batch normalization layers with spectral normalization layers based on WGAN's discriminator and adds a spectral normalization layer after the last convolutional layer. The architectures of discriminators of DCGAN, WGAN, and spectral normalization WGAN are shown in Fig.2(a), Fig.2(b), and Fig.2(c) respectively.



(a) DCGAN Discriminator Architecture



(b) WGAN Discriminator Architecture



(c) Spectral Normalization WGAN Discriminator Architecture

Fig. 2. Discriminator Architecture

4.2. Training methodology

The GAN training algorithm involves training both the discriminator and the generator model in parallel. In the following, training process for each type of GAN model is explained in more details.

4.2.1. Deep Convolutional GAN

Training process in DCGAN networks is generally done in two main steps. First, the discriminator is trained on the real data samples and the error is calculated (training actually produces a probability, that will be used in the binary cross-entropy loss function and compared with the correct label). Basically, the discriminator is first trained to classify real samples from fake samples and get to know how a real sample look like. After that, a batch of fake samples are generated using a random noise passed to the generator. At this time the discriminator is trained on the fake generated samples in order to know how a fake sample look like and again the error is calculated. After this, back-propagation is done and the weights of the discriminator are updated. In the second step, again a random noise is passed to the generator, a batch of fake samples are generated. Theses fake samples are fed to the discriminator (this time without any real samples) The discriminator makes a prediction that produces a probability of how fake these samples are. These probabilities are computed in the binary cross-entropy loss function to compare against the real label, because the generator scores when the discriminator classifies those generated samples as real. With the error rate, we will back-propagate and update the generator's parameters. The above steps performed once in each epoch during training process.

4.2.2. WGAN

WGAN is an extension of GAN in which instead of training a discriminator, a critic is trained which scores the realness or fakeness of a given sample.Training process in WGAN is basically more stable and easier than DCGAN. Training the WGAN is less sensitive to the model architecture and choice of hyper parameters, also the loss of the discriminator apparently is more relatable to the quality of the generated image by the generator.The overall training process is pretty similar to DCGAN but with some differences. The main difference is using the Wasserstein loss function (calculating Wasserstein distance) to train the critic. Another difference is in the back-propagation part, on DCGAN models both the discriminator and the generator weights are updated once in each iteration but in WGAN the discriminator (critic) is trained more times. In our case we updated the weights of the generator every 5 iteration. In WGAN, gradient clipping is also used for the critic model to enforce a Lipschitz constraint. Gradient clipping is basically Constrain critic model weights to a limited range after each mini batch update (e.g. [-0.01,0.01]). Also, Class labels for WGAN is -1 for real images and +1 is for fake images which makes sense due to the fact that there's a critic which scores the realness/fakeness of an image.

4.2.3. WGAN with Spectral normalization

As mentioned before, training GAN networks is not very stable and using weight clipping is not the optimal way to solve this problem. One of the techniques used to resolve this problem is using a weight normalization technique called 'spectral normalization'. Spectral normalization has the convenient property that the Lipschitz constant is the only hyperparameter to be tuned. Normalizing the layers with spectral normalization ensures that the Lipschitz constant for each layer and for the whole network is equal to one. With Spectral Normalization, the weight can be renormalized whenever it is updated. In this way, gradient explosion problems can be mitigated[5]. The rest of the training process is the same as it was before.

5. EXPERIMENTS

5.1. Metrics

While the loss for both the generator and discriminator are logged over time during training for each model (see Figure 4 and 5 below), it is expected that this loss is not enough to evaluate the performance of each model relative to eachother. This is both due to the fact that loss functions may not converge in a min-max game (in the case of the DCGAN) causing highly unstable return values, but also because the loss values between models are not directly comparable. To solve this problem, The Fréchet Inception Distance (FID) was chosen as a metric to validate the results from training, and allow for direct comparison between the models. In the most basic sense, the FID attempts to evaluate an image based on how 'real' it would be perceived to be, by a human evaluator. It is an extension of the Inception Distance (IS), which uses a pre-trained InceptionV3 model to get the conditional label distribution of the image outputs of a GAN, and then assumes that meaningful images would have distributions with low entropy, and that image outputs from a model should be varied[1]. The FID takes this one step further, by also including a 'groundtruth' measure, against distributions of a set of real images, such that the final metric is defined as:

$$\text{FID} = \|\mu_r - \mu_g\|^2 + T_r(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{\frac{1}{2}}) \qquad (3)$$

Where 'r' and 'g' refers to the 'real' and 'generated' image distributions respectively. Since the FID has been shown to better reflect human visual perception of images than IS due to this additional measure [6], the IS has not been included in this study.

5.2. Training and validation

We trained our models on different datasets, several times to validate our results. Each data set (MNIST, CIFAR10, Fashion MNIST) has been trained 10 times on each model. The training process for each dataset is as follows:

- CIFAR training process: 40000 iterations. Calculate FID scores every 200 iterations.
- MNIST and FashionMNIST training process: 10000 iterations. Calculate FID scores every 200 iterations.
- DCGAN training process: training generator and discriminator once every iteration. Optimizer: Adam optimizer with 1e-4 learning rate. Loss: BCELoss.
- WGAN training process: training generator and discriminator once every iteration. Optimizer: Adam optimizer with 1e-4 learning rate. Loss: Earth-mover distance. The weight clipping method is implemented.
- Spectral Normalization WGAN training process: training generator once and discriminator 5 times every iteration. Optimizer: RMSprop with 5e-5 learning rate. Loss: Earth-mover distance.

It is worth mentioning that we normalized all the datasets to the valid normalization values found online. The values can be found in the data loader files in the code on the repository of this report.

6. RESULTS

The changes made to each model impacted not only the quality of their output, but also the ability to make accurate predictions about model performance and potential problems, based on loss values. Figure 3 shows a small selection of generated images for each model and dataset. All three models perform equally well on the 'simpler' datasets (MNIST and FMNIST), based on visual inspection. For CIFAR however, there are some changes between models. DCGAN appears to perform the best, with visually diverse images at a reasonable level of detail. However, WGAN and Spectral Normalisation WGAN both seem to have some amount of overexposure happening in the generated images - the former more so than the latter. Additionally, Spectral normalisation WGAN also displays a pixel-like pattern, causing images to appear less sharp and somewhat distorted.

Looking at the loss graphs for each model (see Figure 4 and 5), the effect of the choice in divergence function becomes immediately clear. The Generator and Discriminator Losses for the DCGAN do indicate whether the model is functioning and has not experienced vanishing gradients (in the case of Figure 4 (a) it has not), but not how the model is performing. However, the EM distance implementation for the WGAN (see

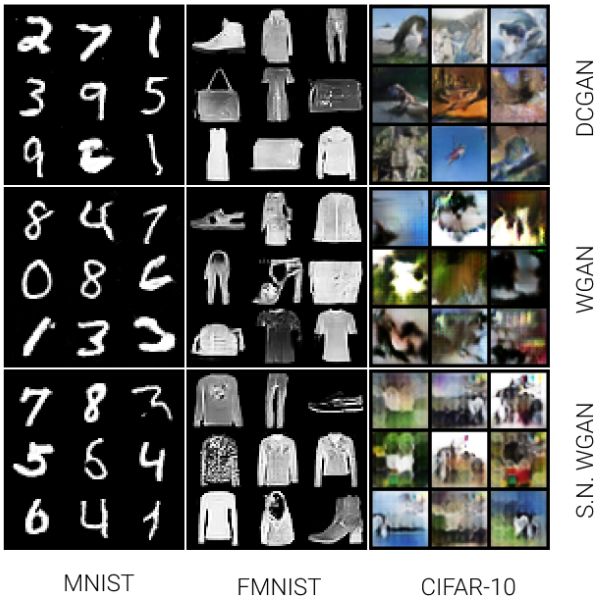


Fig. 3. Image outputs from each model and dataset

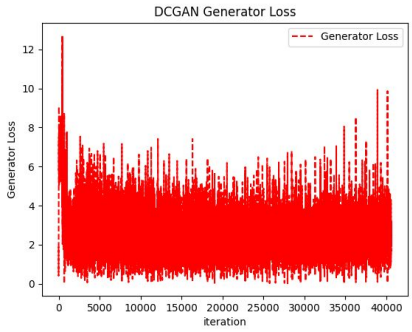
Figure 4 (b) and 5 (b)) clearly shows a degree of convergence, despite the continued amount of fluctuation caused by the 1-Lipschitz continuity not being reliably satisfied. Finally, the addition of the Spectral Normalisation measure (see Figure 4 (c) and 5 (c)) can be seen to reliably satisfy the 1-Lipschitz continuity, and therefore finally works as a more reliable measure of actual model performance. As such, we were able to verify that this model converges and does not suffer from significant mode collapse, as this would appear on the Spectral Normalisation curve as (more) significant fluctuations in loss value.

The average FID score at the time of convergence was calculated and plotted for each model and dataset, as seen in Figure 6. Surprisingly, all models appear to perform better for this metric on the more complex CIFAR dataset, than MNNIST and Fashion MNIST. This is unexpected, because it is contrary to what was observed in Figure 3 for the image sampled at a late iteration, where MNIST and FMNIST appeared to visually outperform CIFAR in quality. These FID scores also suggest, that despite these variations between datasets, WGAN appears to be the best model in terms of image output quality, as it outperforms the others for MNIST and CIFAR, and is a very close second to S.N. WGAN on FMNIST.

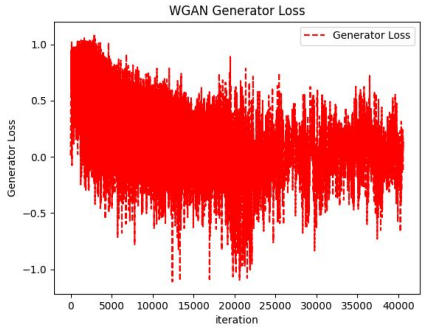
This can also be observed on the average FID score calculated over time (see Figure 7), where WGAN significantly out-performs DCGAN and S.N. WGAN on this metric.

7. DISCUSSION

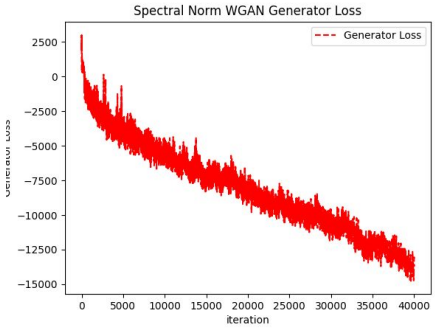
While the results suggest that WGAN is the better model of the three tested, in theory, Spectral Normalization WGAN



(a) DCGAN Generator Loss for CIFAR-10

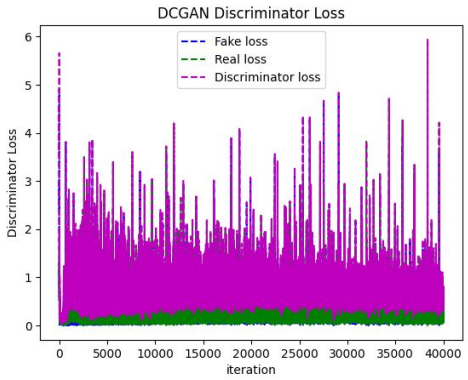


(b) WGAN Generator Loss for CIFAR-10

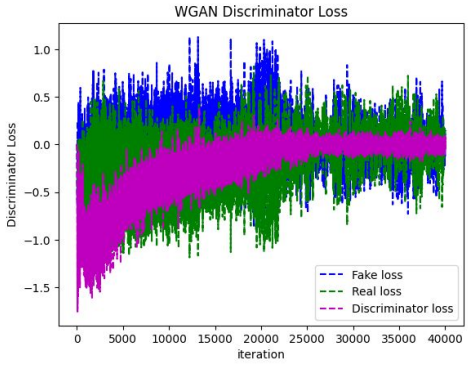


(c) WGAN with Spectral Norm. Generator Loss for CIFAR-10

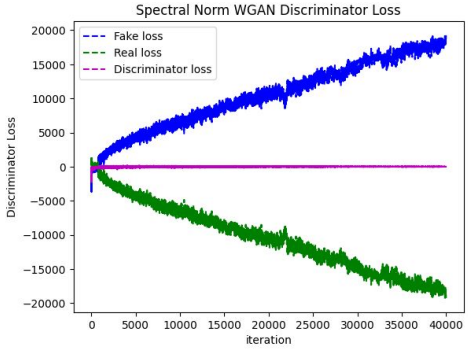
Fig. 4. Generator Loss



(a) DCGAN Discriminator Loss for CIFAR-10



(b) WGAN Discriminator Loss for CIFAR-10



(c) WGAN with Spectral Norm. Discriminator Loss for CIFAR-10

Fig. 5. Discriminator Loss

should get the best results, because it makes the discriminator perfectly satisfy the 1-Lipschitz continuity condition and reliably converges. This suggests that there may be an implementation issue in the architecture of this model which could be worth investigating in the future. Additionally, the higher FID scores for the simpler datasets, despite having observed good image quality from these models, puts into question whether the way the FID score is calculated is correct as well, but could also be a reflection of the relative instability of the models when trained on datasets this simple. Further investigation into this issue would be required in order to determine an exact cause, and confirm the results.

8. CONCLUSION

Based on the results, WGAN and Spectral Normalization WGAN are more stable than DCGAN on simple datasets, while WGAN and DCGAN perform better than Spectral Normalization WGAN on more complex datasets on stability. As WGAN also performs better on MNIST and CIFAR in terms

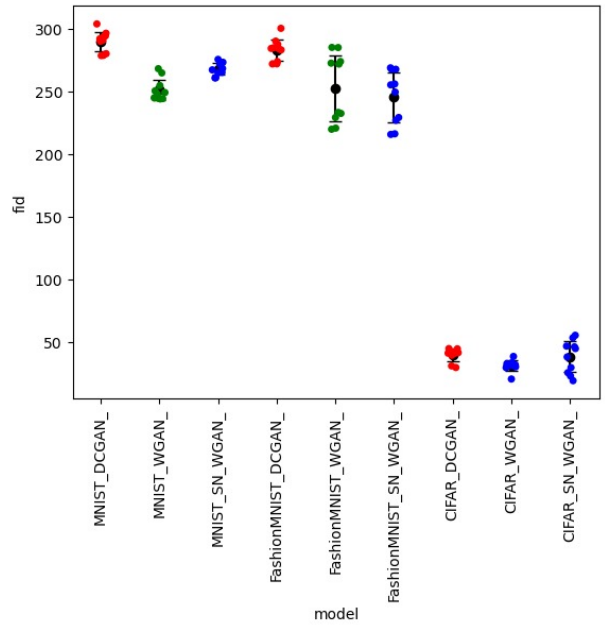


Fig. 6. FID Scores for each model and dataset

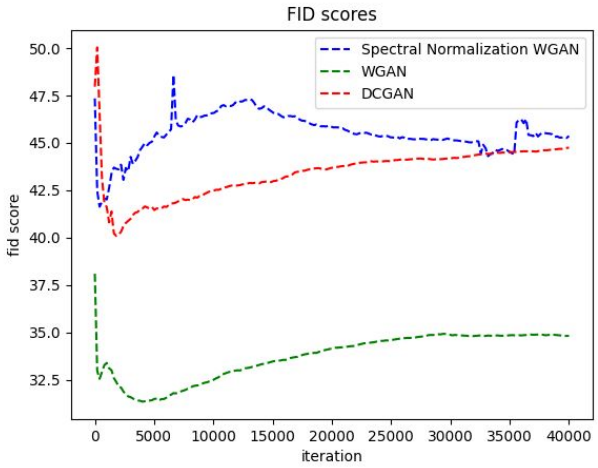


Fig. 7. FID Scores over time for CIFAR-10

of generated image quality, it can be concluded that it is the best-performing model of the three.

9. NOTE FROM THE AUTHORS

After the presentation of these models on the 8th of December 2022, it was found that there was an error in the way the generator was implemented. This error was rectified, and the results were redone in order to reflect this change. This means that the results and conclusions seen here are **not** consistent with those presented previously, and therefore will deviate somewhat from those seen on the poster attached to this report.

10. GIT REPOSITORY

https://github.com/Karl-Liu-ch/Pytorch$_W GAN$

11. REFERENCES

[1] 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*, 2014, pp. 2672–2680.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou, "Wasserstein gan," 2017, cite arxiv:1701.07875.

[3] Jie Gui, Zhenan Sun, Yonggang Wen, Dacheng Tao, and Jieping Ye, "A review on generative adversarial networks: Algorithms, theory, and applications," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2021.

[4] Saifuddin Hitawala, "Comparative study on generative adversarial networks," 2018.

[5] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida, "Spectral normalization for generative adversarial networks," 2018.

[6] 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," in *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. 2017, vol. 30, Curran Associates, Inc.