Evaluating the distribution learning capabilities of GANs

Anonymous Authors[1]

Abstract

We evaluate the distribution learning capabilities of generative adversarial networks by testing them on synthetic datasets. The datasets include common distributions of points in R^n space and images containing polygons of various shapes and sizes. We find that by and large GANs fail to faithfully recreate point datasets which contain discontinous support or sharp bends with noise. Additionally, on image datasets, we find that GANs do not seem to learn to count the number of objects of the same kind in an image. We also highlight the apparent tension between generalization and learning in GANs.

1. Introduction

Generative Adversarial Models (GANs)(Goodfellow et al., 2014) have been found to produce images of very high quality on some datasets (Karras et al., 2018a;b). However, their results on other datasets, while impressive, still lag behind (Brock et al., 2018). This raises the question whether GANs are indeed the right choice to model some distributions. This paper aims to test the distribution learning ability of GANs by evaluating them on synthetic datasets.

1.1. Related works and Contributions

It has been proposed in recent work that "a high number of classes is what makes ImageNet (Deng et al., 2009) synthesis difficult for GANs" (Odena et al., 2017). Indeed, GANs have been able to produce very high quality images on CelebA (Karras et al., 2018b;a) while results on Imagenet are not so impressive (Brock et al., 2018). Because distributions of natural images are complex, in this work, we focus our attention on synthetically generated datasets. We study the learnability of commonly encountered distributions in low dimensional space and the impact of discontinuity. Additionally, we evaluate a specific aspect of learning high dimensional image distributions, counting similar objects in a scene. This constitutes an important part of learning latent space representations of images since for an image to be semantically well-formed, certain objects must obey certain numerical constraints (for example, number of heads on an animal).

Our evaluation is performed on synthetic point and image datasets. To our knowledge, the only instance of synthetic image datasets used for GAN evaluation have been to learn manifolds of convex polygons (specifically triangles) (Lucic et al., 2018). Although, we also use polygons as a testbed for our experiments, we focus on learning a manifold with multiple polygons where their number is fixed.

Our contributions are as follows:

1. We show via experiments on synthetic datasets that commonly found distributions are learnable by GANs. We also highlight that distributions with gaps in support may be difficult to learn without using a mixture of generators.

2. We empirically evaluate whether GANs can learn semantic constraints on objects in images in high dimensional space. Specifically, we test a GAN's ability to count an object that is repeated in an image.

3. We underline a possible tension between generalization ability of GANs and their learning capabilities.

2. Experimental Setup

In this section, we describe the setup of our experiments which include details about the datasets generated, architectures used and the reasoning behind them.

2.1. Datasets

We generate two kinds of datasets (each with 5000 examples) for our evaluation : point datasets, where each sample is a point in R^n and image datasets with each image containing a fixed number of polygons. We use 4 point datasets in our evaluation : Mixtures of Gaussians, Concentric Circles, S-shape curves and Swiss rolls. The first two are 2D while the latter two are in 3D space. This choice was made to enable us to visualize the learned distribution. For each of these four settings, we experimented with three variants, each containing a different amount of noise.

To evaluate high dimensional learning, we generate image datasets which are mixtures of polygons. We created three datasets with each image containing: 1 square of size 4x4 (called Squares 1-4), 3 squares of size 4x4 (called Squares

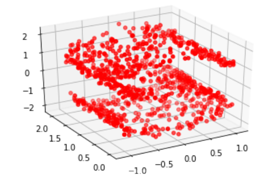

Figure 1. S-Curve Distribution

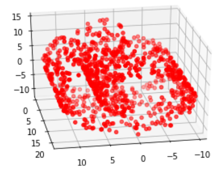

Figure 2. Swiss Roll Distribution

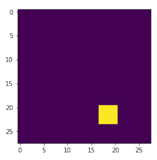

Figure 3. Image from the Squares 1-4 dataset

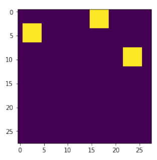

Figure 4. Image from the Squares 3-4 dataset

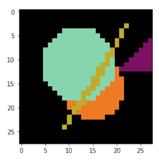

Figure 5. Image from the CT2 dataset

3-4) and a mixture of two triangles and two circles (called CT2). All the datasets contain images of size 28x28 with the third one containing 3 channels. Additionally, for the first two datasets, all squares are non-overlapping and have edges which are axis-aligned. CT2, on the other hand, contains overlapping polygons. In each dataset, the number of objects is fixed and the only varying quantity is their position (which varies with a Gaussian distribution). For the square datasets, even the rotation and shape of the objects is held constant. Hence, the only source of variation is their position. Some examples are shown in Figures 3, 4, 5.

The main objective behind creating images with a fixed number of objects was to test whether GANs can learn to count the number of objects in a scene. More specifically, since GANs have shown impressive image generation capabilities on centered image datasets(Karras et al., 2018a), we want to measure whether that performance can transfer to datasets with objects occuring at varying locations in the scene. Since the only varying quantity in the square datasets is the position, we would expect GANs with true distribution learning abilities to be able to produce images with the exact numbers of squares at different positions in the image.

Additional details about our data generation process can be found in the Appendix.

2.2. Architectures

We use two sets of architectures to train our models. For point datasets, we use a Vanilla GAN with a 3 layer MLP for both the generator and discriminator and another model with the same architecture with Wasserstein loss (enforced via gradient penalty) (Gulrajani et al., 2017).

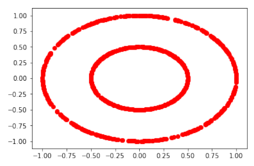

Figure 6. Original distribution of Concentric circles

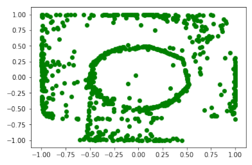

Figure 7. Learned Distribution after 150k steps. **Note:** this distribution gets better after more iterations but we show the one after 150k for homogeneity

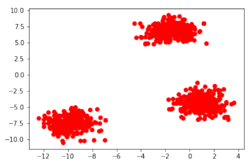

Figure 8. Original distribution of 3 blobs

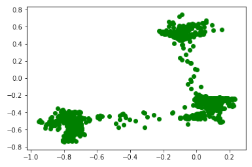

Figure 9. Learned Distribution after 150k steps

For image datasets, we use one model with a DCGAN-inspired (Radford et al., 2016) architecture and another model with the same architecture with Wasserstein loss (enforced via gradient penalty). We do not evaluate Vanilla GANs on our image datasets because they do not seem to be competitive with the other models in our experiments. Further architectural details (including choice of hyperparameters) are described in the Appendix.

2.3. Experimental Details

We used the Google Colaboratory environment (12GB RAM Nvidia Tesla K80) for all our experiments in this paper. All models with Wasserstein loss are trained with RMSProp (Tieleman & Hinton, 2012) with a learning rate of 0.00005. All others are trained with the ADAM (Kingma & Ba, 2015) optimizer with a learning rate of 0.0002. We train models for up to 150k training steps and stop earlier if the model reaches convergence earlier. Increasing training steps beyond 150k were not found to significantly improve sample quality. For each generator update, we update the discriminator 5 times for WGAN-GP inspired architectures. Additional details about our experiments and architectures can be found in the Appendix.

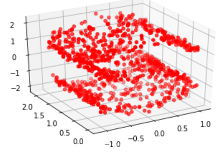

Figure 10. Original distribution of the shape S (minimal noise)

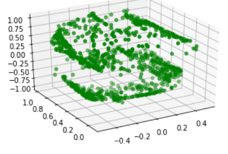

Figure 11. Learned Distribution after 140k steps (minimal noise)

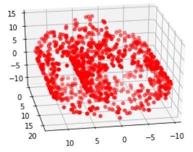

Figure 12. Original distribution of the Swiss roll (minimal noise)

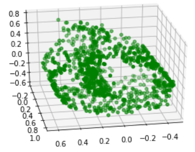

Figure 13. Learned Distribution after 150k steps (minimal noise)

3. Results and Analysis

3.1. Point Data

On mixtures of Gaussians and concentric circles, we find that both architectures seem to perform equally well. They seem to approach approximate convergence as both their discriminator's accuracy oscillates around 50%. Since both datasets contain disconnected components, we find that both models are not able to model this discontinuity and as a result still produce samples which lie in between clusters of data. This may explain the oscillatory behaviour observed. Examples of 1000 samples from the real and fake distributions are shown in Figures 7, 9.

The inability to model discontinuity is understandable since the latent space is continuous and the neural network can be considered to be a continuous function approximator so the output has to be continuous as well. This virtually guarantees that some samples from the model will be necessarily "bad".

Next, we evaluate the S-curve and Swiss roll distributions. Traditionally, mixtures of Gaussians have been the toy distribution of choice for GAN evaluation. However, we find that both distributions are learned fairly faithfully but increasing noise can cause separate surfaces in the distribution to be merged. For example, with increased noise, the S shape in the S-curve can become an 8 or the Swiss roll may look like a circle (samples in Appendix).

This phenomenon, in our experiments, seems to be worse (in terms of losing shape) for Swiss rolls than S-curves. We hypothesize this may be due to overlapping surfaces being closer in the Swiss roll distribution making it more sensitive to noise. This dependence of sensitivity to noise on the underlying distribution suggests that GANs may not be suitable to modelling certain distributions in noisy environments and alternative generative models may need to be explored.

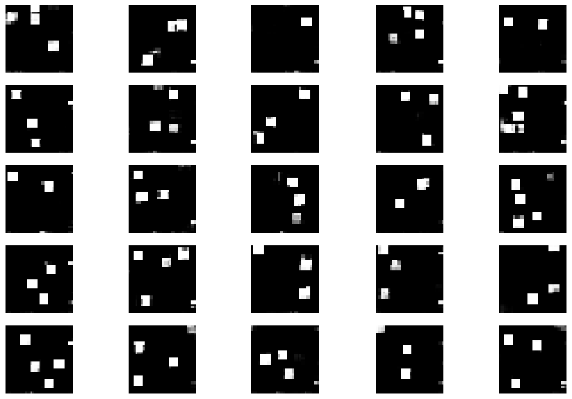

Figure 14. Samples after 150k steps from DCGAN (converged)

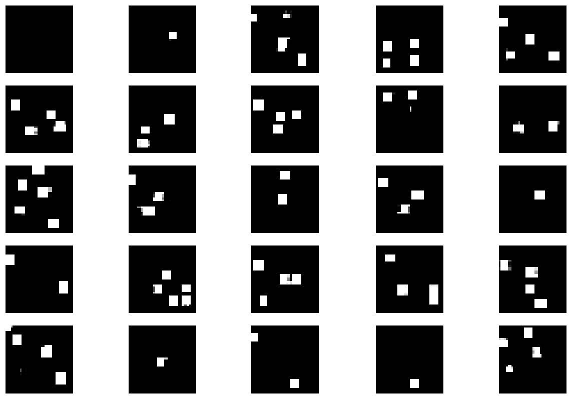

Figure 15. Samples after 150k steps from WGAN-GP

3.2. Image Datasets

During training, the first model (without Wasserstein loss) seems to converge after about 74k steps while the other one doesn't seem to converge. Inspite of this difference, we do not find a major difference in image quality from the two models after 150k steps.

We find that both models fail to learn to count on the Squares 3-4 dataset. Our models can potentially produce anything between 0 and 5 squares. Some random samples can be seen in 14, 15.

This raises an important question about the learning capability of GANs. Is what we are seeing just poor learning or should it be instead viewed as generalization? For example, in natural image datasets, say faces, we might see GANs produce images with completely new hair placement. When these images look ”natural” to the observer, they might see it as generalization. However, when that image does not look plausible, we would call it poor learning. As it stands, there seems to be no clear demarcation between poor learning outcomes and generalization.

Since the only source of variation in our image datasets is location, we would like to see the GAN learn that there are 3 squares in each image and then generalize their position. However, currently, we have no way of enforcing this constraint on the number or shape of objects. Quantifying this apparent tradeoff between learning and generalization is an interesting avenue for further work. In fact, there might not even be tradeoff and GANs may fundamentally be unable to learn distributions of such a nature. We leave the evaluation of these hypotheses as future work.

We do not focus our discussion excessively on shape because for more natural datasets, shape is more variable and is not rigid, like in our case. For example, we know that cats have 4 legs, but, the legs may not look the same in every image. A natural objection would be that a similar reasoning would be applicable to the count of objects i.e. some of these legs may be occluded in the image and, therefore, it may appear to a GAN that a cat could have 3 legs. Thus, GANs producing cats with less than 4 legs could be justified.

That is precisely why we have chosen non-overlapping squares for our experiment. Since there are no occlusions, the GAN should learn that in every image, there are supposed to be exactly 3 squares. Even in real world datasets, it may not be possible to collect a dataset which has cats whose legs have the same shape. However, it is possible to collect a dataset with cats having four legs in all images (no occlusions).

Interestingly, both models seem to learn that there is only one square in the 1-4 dataset i.e. when trained on the Squares 1-4 dataset, most samples seem to include just 1 square of size which visually looks close to 4x4. Admittedly, this visual test of similarity may not be enough to ascertain whether the shape and size of the square corresponds exactly to a 4x4 square. However, during training with images of 1 square, we observed that if we increase the size of the square i.e. instead of using 4x4 use 16x16 (Squares 1-16), the likelihood of having multiple squares in the image reduces.

We also observe that all squares generated seem to be axis aligned. Therefore, GANs seem to have no problem learning orientation but cannot seem to enforce counting constraints.

We also investigated whether a GAN can leverage the fact that it has learnt what a square looks like on Squares 1-4. In this experiment, we transferred the weights from that trained model and fine-tuned it on the Squares 3-4 dataset. We did not observe a noticeable improvement in image quality. If anything, image quality tends to get worse.

Samples from models trained on CT2 and additional samples from our models are included in the Appendix. We observed that the GANs did not sufficiently reproduce circles and triangles when trained on CT2. Since CT2 contains overlaps and multiple types of polygons, we consider this to be a more challenging dataset to model.

4. Conclusion

In this paper, we present the phenomenon of GANs being unable to count. We support this hypothesis with experiments on synthetic datasets where the count of similar objects in a scene is kept constant while their location is varied. We find that in their current form GANs are unable to learn semantic constraints even in the absence of noise introduced by natural image datasets. We also emphasize the fine line between generalization and good learning outcomes in GANs. Additionally, we conduct experiments on non-image data where we conclude that GANs tend to have difficulty learning discontinuous distributions which might necessitate the usage of mixtures of generators. A thorough evaluation of such an approach is left as future work.

References

Bradski, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

Brock, A., Donahue, J., and Simonyan, K. Large scale GAN training for high fidelity natural image synthesis. *CoRR*, abs/1809.11096, 2018.

Chintala, S. How to Train a GAN? Tips and tricks to make GANs work. https://github.com/soumith/ganhacks, 2016. [Online; accessed 1-May-2019].

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pp. 2672–2680, Cambridge, MA, USA, 2014. MIT Press. URL http://dl.acm.org/citation.cfm?id=2969033.2969125.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. Improved training of wasserstein gans. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 5769–5779, USA, 2017. Curran Associates Inc. ISBN

978-1-5108-6096-4. URL http://dl.acm.org/citation.cfm?id=3295222.3295327.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of gans for improved quality, stability, and variation. In *ICLR*. OpenReview.net, 2018a.

Karras, T., Laine, S., and Aila, T. A style-based generator architecture for generative adversarial networks. *CoRR*, abs/1812.04948, 2018b.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Lucic, M., Kurach, K., Michalski, M., Bousquet, O., and Gelly, S. Are gans created equal? a large-scale study. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 698–707, USA, 2018. Curran Associates Inc. URL http://dl.acm.org/citation.cfm?id=3326943.3327008.

Odena, A., Olah, C., and Shlens, J. Conditional image synthesis with auxiliary classifier GANs. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2642–2651, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/odena17a.html.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016.

Tieleman, T. and Hinton, G. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

A. Data Generation

Here we describe the algorithms used for dataset creation.

A.1. Point Datasets

We used scikit-learn's (Pedregosa et al., 2011) sklearn.datasets module for dataset creation. Each dataset has 5000 examples and consists of points in R^2 or R^3 for easier visualization.

Dataset	Function used	Dimension
Circles	make_circles	2
Mix of Gaussians	make_blobs	2
S Curve	make_s_curve	3
Swiss Roll	make_swiss_roll	3

Table 1. Point Dataset summary

Each dataset has associated noise parameters corresponding to the noise parameters in the scikit-learn API. We experiment with varying noise but we find that it does not affect learning outcomes too much.

A.2. Image Datasets

The Squares datasets consist of non overlapping squares to test whether GANs can learn to count and to avoid introducing noise in the learning stage. To create images, we sample 3 random points in the image, which will serve as the top left point of the square. Then, we check whether squares drawn at these point overlap or not. If they do, then we again sample 3 new points otherwise we draw the squares to the image and add it to the dataset.

For the CT2 dataset, we use OpenCV's (Bradski, 2000) draw.ellipse() and draw.polygon() functions to draw figures at random point in the image. Note circles and traingles can overlap in this dataset, therefore, we consider this to be a more challenging dataset for a GAN to model.

B. Architectural Details

The architectures used in our experiments are given in the Tables 2, 3, 4 and 5. We chose our architectures in accordance with common guidelines in GAN architectures (Radford et al., 2016; Gulrajani et al., 2017; Chintala, 2016). Minor changes in the architecture (e.g changing ReLUs to Leaky ReLUs in the generator, adding or removing a few layers) did not seem to matter in terms of image quality in our experiments.

The size of the latent space for the image datasets was chosen to be 100 and for point datasets, it was chosen to be 2 for points in R^2 and 3 for points in R^3.

C. Additional Samples

Additional Samples from our models are shown in Figures 16, 17, 18, 19, 20, 21, 22, 23, 24.

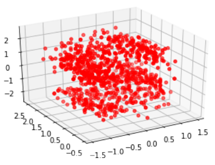
Figure 16. Original distribution of the S Curve (extra noise)

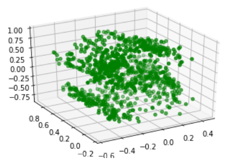
Figure 17. Learned Distribution after 150k steps (extra noise)

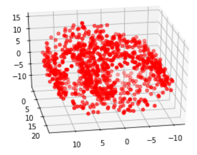
Figure 18. Original distribution of the Swiss Roll (extra noise)

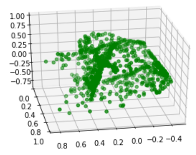
Figure 19. Learned Distribution after 150k steps (extra noise)

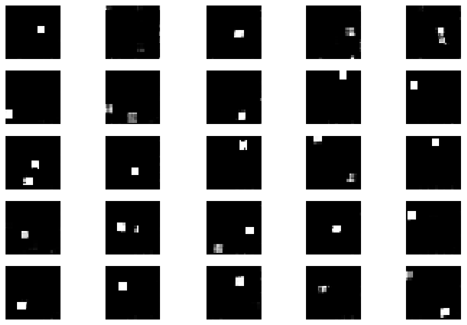
Figure 20. Samples from model trained on Squares 1-4

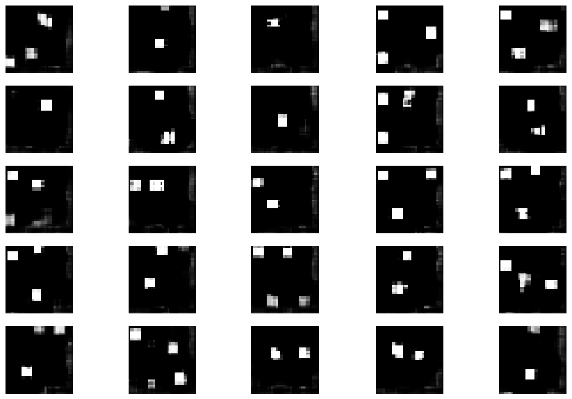
Figure 21. Samples from models with transfer from Squares 1-4 to Squares 3-4

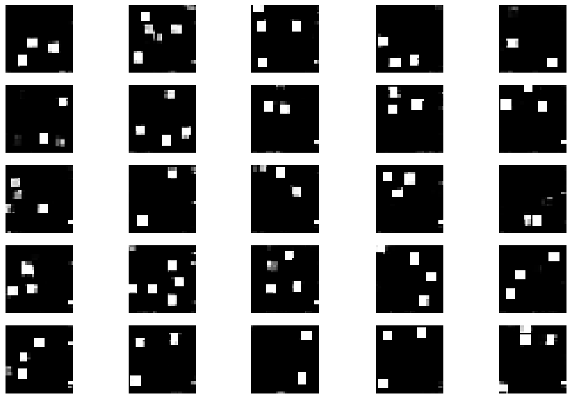
Figure 22. Additional samples from DCGAN

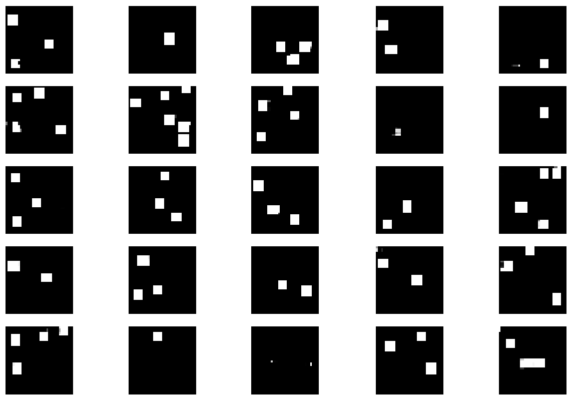
Figure 23. Additional samples from WGAN-GP

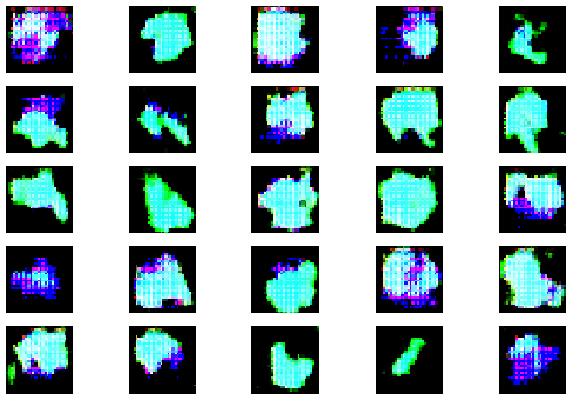
Figure 24. Samples from model trained on CT-2 after 150k iters

Generator	Discriminator
Dense	Dense
Leaky ReLU	Leaky ReLU
Dense	Dense
Leaky ReLU	Leaky ReLU
Dense	Dense
Leaky ReLU	Leaky ReLU
Dense	Dense
Tanh	Sigmoid

Table 2. Vanilla GAN Architecture for Points

Generator	Discriminator
Dense	Dense
Leaky ReLU	Leaky ReLU
Dense	Dense
Leaky ReLU	Leaky ReLU
Dense	Dense
Leaky ReLU	Leaky ReLU
Dense	Dense
Tanh	-

Table 3. WGAN-GP Architecture for Points

Generator	Discriminator
Dense	Conv2D
Leaky ReLU	Leaky ReLU
Reshape	Dropout
Transposed Conv2D	Conv2D
Batch Norm	Batch Norm
Leaky ReLU	Leaky ReLU
Transposed Conv2D	Dropout
Batch Norm	Conv2D
Leaky ReLU	Batch Norm
Transposed Conv2D	Leaky ReLU
Tanh	Dropout
-	Conv2D
-	Batch Norm
-	Leaky ReLU
-	Dropout
-	Sigmoid

Table 4. DCGAN Architecture for Images

Generator	Critic
Dense	Conv2D
ReLU	Leaky ReLU
Reshape	Dropout
Transposed Conv2D	Conv2D
Batch Norm	Batch Norm
ReLU	Leaky ReLU
Transposed Conv2D	Dropout
Batch Norm	Conv2D
Leaky ReLU	Batch Norm
Transposed Conv2D	Leaky ReLU
Tanh	Dropout
-	Conv2D
-	Batch Norm
-	Leaky ReLU
-	Dropout
-	Dense

Table 5. WGAN-GP Architecture for Images