Lost in Translation: GANs' Inability to Generate Simple Probability Distributions

Debanjan Dutta, Anish Chakrabarty*, and Swagatam Das
Electronics and Communication Sciences Unit, Indian Statistical Institute, Kolkata
*Statistics and Mathematics Unit, Indian Statistical Institute, Kolkata

Abstract

Since its inception, Generative Adversarial Networks (GAN) have marked a triumph in generative modeling. Its impeccable capacity to mimic observations from unknown probability distributions has positioned it as a widely used simulation tool. In typical applications, GANs find themselves simulating data rich in semantic information such as images or text out of random noise. As such, it is reasonable to expect that large parametric models such as GANs must be able to estimate standard theoretical probability densities with ease. In this paper, based on a series of disillusioning experimental findings, we show that GANs often fail to induce the simplest of statistical transformations between distributions. For example, starting with a standard Gaussian noise, GANs with 2-deep generators are unable to perform a positional translation. Supporting theoretical tests on generated data further corroborates our rather unsettling conclusions.

1 Introduction

Statistical simulations often deal with the problem of generating samples from a probability distribution given random noise. Perhaps the most rudimentary method in this regard is the inverse transform sampling. It is based on the principle that given $u \sim U(0,1)$, we have $F_X^{-1}(u) =_d X$, where F_X^{-1} denotes the generalized inverse of the cumulative distribution function of the random variable X. Modern-day generative models deploying deep neural networks (NN), characterized by large sets of parameters ($\Theta \subset \mathbb{R}^m$), essentially run on the same philosophy. Their goal lies in producing pseudo-random replicates from a target distribution $F_X(\cdot)$, often *assumed* to possess corresponding density $f_X : \mathcal{X} \to \mathbb{R}$. Typically, based on the complexity of underlying usages (e.g. generating image samples), such distributions remain unknown to practitioners. Thus, the problem boils down to finding the best estimate of f_X amongst the class of generated laws $g_\theta(\cdot)$, $\theta \in \Theta$. In a GAN (Goodfellow et al., 2014) architecture, the *generator* plays the role of $g_\theta(\cdot)$ and ideally transforms input u to near-replicates of X.

Data samples such as images or text are complex mathematical structures (e.g. high-dimensional vectors, tensors, or graphs), comprised of numerous features that encapsulate semantic information about their parent distribution. As such, estimating at least some of them efficiently (based on learned representations) results in perceptually acceptable simulations. As a result, the complexity of Θ in deep generative models such as GANs is made purposefully high. It enables the class of functions $g_\Theta(\cdot)$ to be rich enough to produce a good estimate. Predictably, GANs have been shown to excel at estimating Besov densities (Liang, 2021). In other words, GANs tend to perform well when the target distribution is regular or has intrinsic 'patterns'. Thus, it is natural to ask whether there exists a lower limit to the amount of semantic information a target law needs to possess to be estimated accurately by GANs. From a simulation perspective, this translates to checking whether GANs can generate samples from standard probability distributions that are exactly characterized by a sufficiently small set of parameters. When seen from a model selection viewpoint, the answer seems straightforward. An over-parameterized model, upon adequate training, should ideally perform well given a problem of lower complexity. However, GANs and their immediate variants exhibit poor approximation in case the target distribution is not information-rich. We elucidate the idea based on the following experiments. Codes and supporting material can be found in `https://github.com/DDuttaGit/ICLRTP2024`.

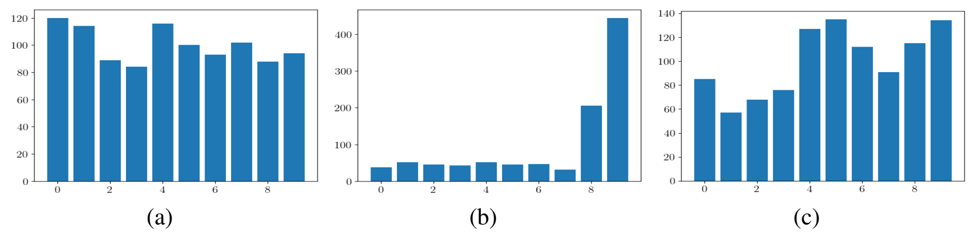

Figure 1: The histograms corresponding to generated distributions using (a) vanilla GAN, (b) WGAN, and (c) LSGAN on input $U(0, 1)$; all deploying leaky-ReLU-activated 2-deep generators.

2 Experiments

We device three experiments inspired by classical statistical transforms. Firstly, we aim to simulate $g_\theta(u) \sim N(0, 1)$ out of $u \in U(0, 1)$. Box-Muller transform (Box & Muller, 1958) readily provides a deterministic pathway based on radial maps. We check whether GANs can induce such functions as well. The architectures deployed in the process are namely vanilla GAN, WGAN (Arjovsky et al., 2017), and LSGAN (Mao et al., 2016). All of such models are constructed using 2-deep generator (G) and discriminator (D) neural networks. We test the effect of multiple activation functions (ReLU, leaky-ReLU, and tanh) under this regime. The choice of the number of hidden layers in G is inspired by the fact that such networks are optimal at approximating radial functions (Eldan & Shamir, 2016). Also, G, in its final layer uses $\tanh(\cdot)$ activation exclusively to span the support.

Table 1: Tests of Normality on $g_{\theta^*}(u)$.

Architecture	AD test	KS test
vanilla GAN	✗	✗
WGAN	✗	✗
LSGAN	✗	✗

The generated distributions under all three models using leaky-ReLU activation exhibit significant departure from Normality [see, Fig 1]. While plotting the histograms, we follow the common practice of standardizing the data and compartmentalizing the range into a substantial number of bins. Standardization seems plausible since our work highlights rather the departure of the generated distributions from Normality. To further examine, we carry out Anderson-Darling (AD) and Kolmogorov-Smirnov (KS) tests of Normality. At 5% level of significance, we observe that both test results reject the hypothesis that $g_{\theta^*}(u)$ is Gaussian [see, Table 1], where θ^* is the resultant parametric value at convergence. The model being overly parameterized, one might suspect that the failure is rooted in overfitting. However, using Dropout regularization (Srivastava et al., 2014), we arrive at the same conclusion [see, Appendix B]. As such, the effect of overfitting, if at all present, is benign. Also, WGAN attests to the finding as it innately clips gradients to enforce convergence.

2.1 Translating and Scaling Gaussians

In this section, we put GANs to the test at one of the simplest tasks in statistical simulation. First, starting with an input $N(0, 1)$ variate, we aim to generate an observation from $N(\mu, 1)$, given $\mu \neq 0$. Ideally, the underlying task involves dynamically finding a near-estimate of μ during training, followed by its addition to the input samples. Observe that, the large parameter set Θ is responsible for sculpting μ. Intuitively, this should be straightforward since obtaining a lower-dimensional estimate can be done solely by introducing sparsity. To distinctly observe the performance of GANs, we specify $\mu = -200$. The last experiment extends this setup by introducing a variance $\sigma^2 \neq 1$. In other words, we aim to generate samples from $N(\mu, \sigma^2)$ using standard Gaussian inputs. Both transformations are easy to realize as feed-forward NNs are essentially affine transforms followed by activations. However, all three of our models (GAN, WGAN, and LSGAN) fail at both. While histograms and QQ plots corresponding to generated samples serve as qualitative evidence, we test $H_0 : (\mu = -200, \sigma^2 = 25)$ against H_1 : 'Not H_0' to testify our findings [see, Appendix B].

3 Conclusion

Our experiments empirically show that despite theoretical assurances, GAN and its variants fail to simulate even the simplest of probability distributions. As a plausible explanation, we point toward the absence of semantic features in standard densities which turn out to be the cornerstone of GANs' capability to recognize. Future work may search for a quantitative measure corresponding to a target distribution that indicates whether GANs can successfully simulate observations from it.

URM Statement

We acknowledge that at least one key author of this work meets the URM criteria of ICLR 2024 Tiny Papers Track.

Acknowledgement

We would like to thank Faizanuddin Ansari, senior research fellow at the Indian Statistical Institute, Kolkata for his valuable remarks. We also appreciate the referees for their pertinent comments.

References

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 214–223. PMLR, 06–11 Aug 2017.

George EP Box and Mervin E Muller. A note on the generation of random normal deviates. *The annals of mathematical statistics*, 29(2):610–611, 1958.

Ronen Eldan and Ohad Shamir. The power of depth for feedforward neural networks. In *Conference on learning theory*, pp. 907–940. PMLR, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Tengyuan Liang. How well generative adversarial networks learn distributions. *The Journal of Machine Learning Research*, 22(1):10366–10406, 2021.

Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2813–2821, 2016.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

Michael A Stephens. The anderson-darling statistic. *No. TR-39). Stanford Univ Ca Dept. of Statistics*, 1979.

A Appendix: Preliminaries

In this section, we provide some rudimentary concepts that are extensively used in our discussion.

Definition A.1 (Feed-forward Neural Networks) *Given $L \in \mathbb{N}^+$, a L-deep Neural Network (NN) is defined as the collection of maps $\phi : \mathbb{R}^{N_0} \longrightarrow \mathbb{R}^{N_{L+1}}$, $\{N_i\}_{i=0}^{L+1} \in \mathbb{N}^+$ given by*

$$\phi(x) := A_L \circ \sigma \circ A_{L-1} \circ ... \circ \sigma \circ A_0(x),$$

where $A_i(y) = M_i y + b_i$; $M_i \in \mathbb{R}^{N_{i+1} \times N_i}$ and $b_i \in \mathbb{R}^{N_{i+1}}$, $i = 0, ..., L$. Here, σ signifies the activation function. Under this setup, we call $W = \max\{N_i\}_{i=1}^{L}$ the width of the network and L its depth.

A.1 GAN loss functions

Let us go through the three loss functions corresponding to the architectures we use to carry out our experiments. Given a generator transform G and discriminator D, the vanilla GAN loss in our setup is given as

$$\mathcal{L}_{\text{GAN}} = \mathbb{E}_x[\log{(D(x))}] + \mathbb{E}_u[\log{(1 - D(G(u)))}].$$

The WGAN loss generalizes the standard Wasserstein distance by taking the supremum over all critic functions induced by the discriminator. As such, the specific form turns out to be

$$\mathcal{L}_{\text{WGAN}} = \sup_D \mathbb{E}_x[D(x)] - \mathbb{E}_u[D(G(u))].$$

The loss function of LSGAN is an exact extension of its vanilla counterpart, replacing only the $\log$ by L_2 norm. The real-valued data used in our experiments motivates us to use LSGAN in particular.

B Appendix: Additional Experiments

Here, we place all experiments from Section 2.1. For the generation process $N(0,1) \overset{G}{\rightarrow} N(-200,1)$, we observe that the quantiles of the generated distribution do not align with the target distribution [see, Fig 4] (after 1000 epochs in vanilla GAN and 200 - 500 for the rest) even when the losses converge [e.g. see, Fig 7(b)]. Training of all models involves data samples of size ranging from $1000 - 5000$. The increasing sample size does not improve the generated quality significantly. Moreover, reducing the complexity of the discriminator function does not improve the generation capacity either. To demonstrate the same, we rerun the experiment with a 1-deep D based on ReLU activations. Here also, the generated distributions show significant departure from $N(-200,1)$ [see, Fig 3]. To statistically validate our findings, we test the equality of means of the two distributions (*real* and *fake*) at 5% level of significance. Table 2 shows all of them getting rejected.

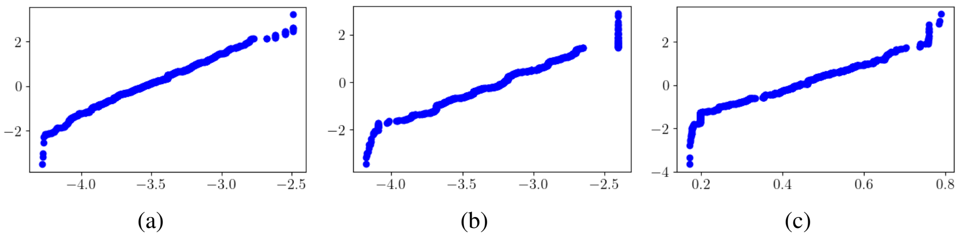

Figure 2: QQ plots corresponding to generated distributions using (a) vanilla GAN, (b) WGAN, and (c) LSGAN on input $U(0,1)$ to achieve $N(0,1)$; all deploying tanh-activated 2-deep generators with Dropout.

Table 2: Tests of equality of parameters between generated and target distributions.

Architecture	$N(0,1) \overset{G}{\to} N(-200,1)$	$N(0,1) \overset{G}{\to} N(-200,25)$
	Test of equality of mean	Tests of equality of mean and variance
vanilla GAN	✗	✗
WGAN	✗	✗
LSGAN	✗	✗

*The decisions 'Accept' and 'Reject' against the null hypotheses are denoted by the symbols (✓) and (✗) respectively.

We carry out the tests under varying sample sizes (500 - 4000). 'Rejection' is reported based on the majority of a fixed number of test results coming out as so. We mention that the critical values corresponding to the underlying tests (AD and KS) were originally tabulated based on the asymptotic distributions of the statistic due to their consistency. As such, they perform well even if the number of samples is large enough. For example, the critical values presented in Stephens (1979) only require the sample size to be ≥ 5 for a one-sample AD test.

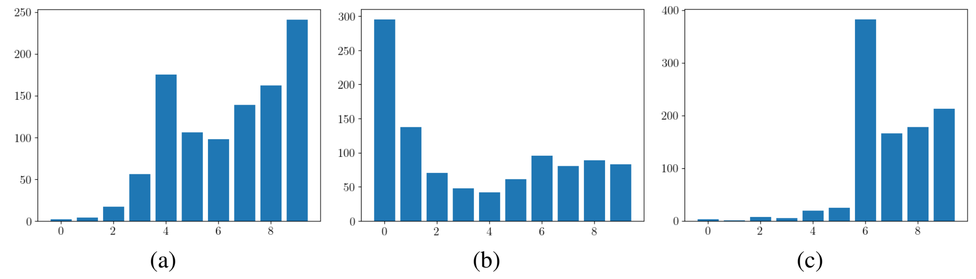

Figure 3: The histograms corresponding to generated distributions using (a) vanilla GAN, (b) WGAN, and (c) LSGAN on input $N(0,1)$ to achieve $N(-200,1)$; all deploying leaky-ReLU-activated 2-deep generators and 1-deep ReLU-activated discriminators.

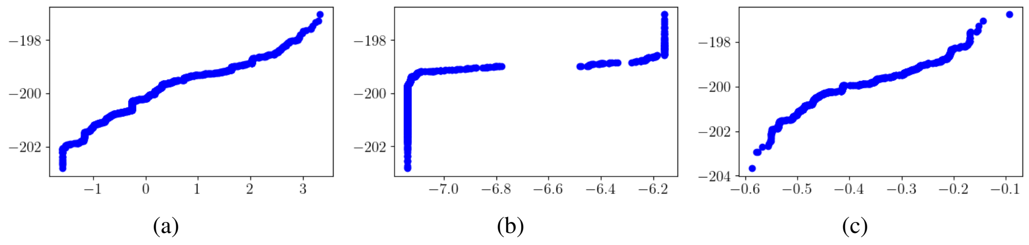

Figure 4: QQ plots corresponding to generated distributions using (a) vanilla GAN, (b) WGAN, and (c) LSGAN on input $N(0,1)$ to achieve $N(-200,1)$; all deploying leaky-ReLU-activated 2-deep generators with Dropout.

Carrying out the same experiments for the process $N(0,1) \overset{G}{\to} N(-200,25)$ bear similar outcomes. The histogram and QQ plots of the generated distribution do not hint at Normality [see, Fig 5, 6], let alone accurate central tendency or dispersion, under satisfactory convergence [e.g see, Fig 7(a) and 8(c)]. To statistically validate, we use tests of equality of (μ, σ), which based on the observed data rejects the null hypothesis at 5% level of significance [see, Table 2]. Such evidence enables us to conclude that GANs, (also WGAN and LSGAN) having the architecture under consideration, fail to generate some of the most simple distributions despite convergence.

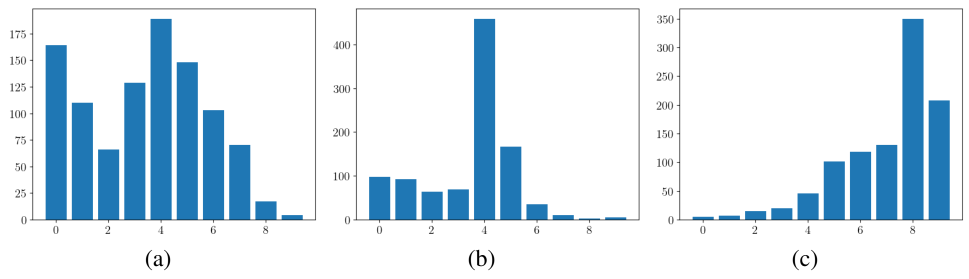

Figure 5: QQ plots corresponding to generated distributions using (a) vanilla GAN, (b) WGAN, and (c) LSGAN on input $N(0, 1)$ to achieve $N(-200, 25)$; all deploying leaky-ReLU-activated 2-deep generators.

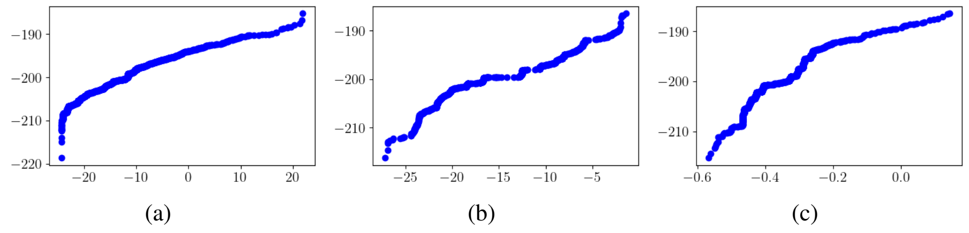

Figure 6: QQ plots corresponding to generated distributions using (a) vanilla GAN (b) WGAN, and (c) LSGAN all deploying Tanh-activated dropped-out 2-deep generators; on input $N(0, 1)$ to achieve $N(-200, 25)$.

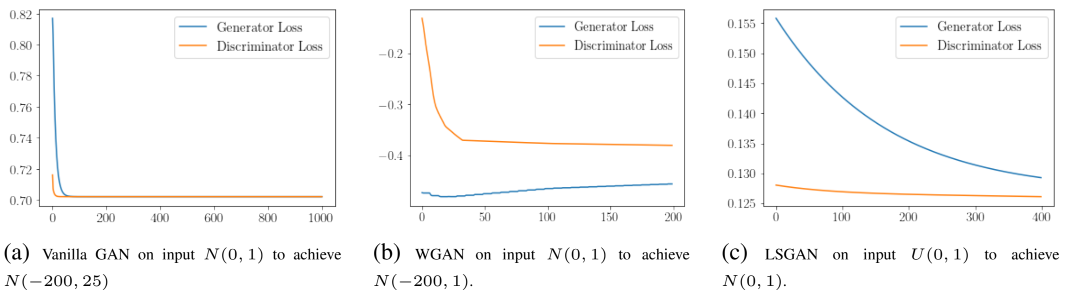

Figure 7: Loss plots of generator and discriminator for different GAN models under different learning rates. Subfigure (a) is using (generator, discriminator) learning rates $(0.0001, 0.005)$ while subfigure (b) and (c) are using $(0.001, 0.001)$.

We have further studied the performance of different GANs on input $U(0, 1)$ to achieve $N(-200, 1)$.

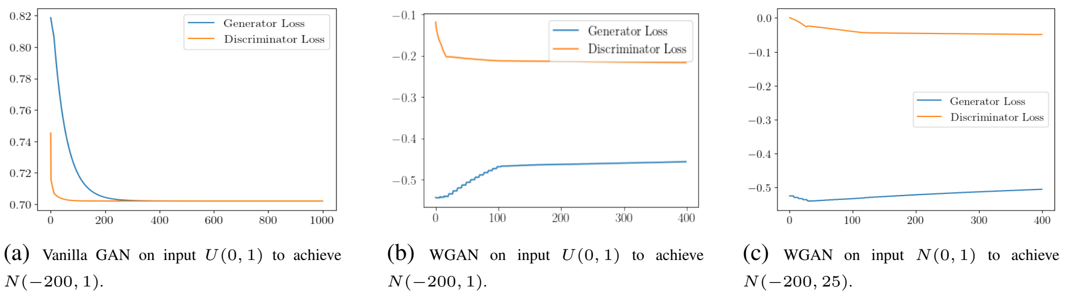

Figure 8: Further set of loss plots of generator and discriminator for GAN variants under different learning rates. Subfigure (a) is using (generator, discriminator) learning rates $(0.0001, 0.001)$, subfigure (b) and (c) are using $(0.001, 0.001)$.