WHAT ARE GANS USEFUL FOR?

Briland Hitaj *
Computer Science Department, Stevens Institute of Technology
bhitaj@stevens.edu

Pablo M. Olmos
Departamento de Teoría de Señal y Comunicaciones, Universidad Carlos III de Madrid
olmos@tsc.uc3m.es

Paolo Gasti
Computer Science Department, New York Institute of Technology
pgasti@nyit.edu

Giuseppe Ateniese
Computer Science Department, Stevens Institute of Technology
gatenies@stevens.edu

Fernando Pérez-Cruz
Swiss Data Science Center, (ETH Zurich and EPFL)
fernando.perezcruz@sdsc.ethz.ch

ABSTRACT

GANs have shown how deep neural networks can be successfully used for generative modeling. As a result, generative models based on neural networks are expected to impact many fields the same way that discriminative modeling based on neural networks did. Early uses of GANs led to impressive results: GANs successfully generated high-quality samples in high dimensional structured spaces, such as images and text, that were present in the training data. However, evaluating generative and discriminative models are fundamentally different tasks. While discriminative learning has a clear goal and a well-defined set of evaluation metrics, generative modeling is an intermediate step towards understanding the data or generate hypothesis. As such, the quality of implicit density estimation is hard to evaluate, because in general it is hard to measure how well a data sample is represented by the model. For instance, how can one assess whether a generative process is generating natural samples, sampled from the appropriate distribution?

In this paper, we take a fresh look at GANs, and at the quality of their output. We observe that even though GANs might not be able to generate samples from the underlying distribution, they can still capture important structure information from data in high dimensional space. Our conclusion is that, in many applications, the ability to perform precise density estimation is not required for the construction of a useful generative model. We consider this work a first step towards determining how estimates produced by GANs can be leveraged to improve on other generative modeling algorithms.

1 INTRODUCTION

Generative Adversarial Models (GANs) (**?**) and implicit generative models (**??**) are very appealing to many areas of science and engineering that rely on large amounts of hard to collect data samples, because they might implement *universal simulators*. For instance a geneticist could train a GAN

*Also, PhD Student at Sapienza University of Rome

using genomic data from hundreds of millions of humans, and then use the GAN to *imagine* the genome of all possible human. Similarly, a climate scientist would use a GAN, trained on years of weather data, to envision any worldwide weather pattern. These simulators are not an *end goal* of their own, no matter how difficult they are to build. Rather, they are *tools* that enable us to understand complex phenomena using data: given a universal simulator we could, automatically and inexpensively, generate the data required to validate (or to reject) any hypothesis.

GANs are based on deep learning techniques, which have been extremely successful in discriminative modeling. Deep learning's success is primarily due to its ability to learn relevant features together with the appropriate classifier. This is in contrast with prior techniques, which rely on low-dimensional intuitively-human-engineered (and therefore limited) features. For instance, Vapnik, criticized the paradigms created by parametric approach, stating that: "*[t]o find a functional dependency from data, the statistician is able to define a set of functions, linear in their parameters, that contain a good approximation to the desired function. The number of parameters describing this set is small*" (**?**). In contrast, deep Learning removes the need to define a priori this set of functions.

Implicit models in discriminative learning are well suited because typically the goal is clear (e.g., finding objects in an image, converting speech into text, or automatic machine translation), and there are well-define metrics for validating the model's results. However, generative modeling does not have such clear goals and metrics. Generative model are always an intermediate step towards solving some other problem. They must understand the available data, and generate (or validate) hypotheses about it. Broadly speaking, we can classify generative models in two groups: (1) those that estimate the density of the data accurately, and can therefore be used for any application; and (2) those that extract a representation that is *actionable* (even though the generative model might not fully capture the data generating process), and target a specific question about the data.

The latter group includes dimensionality reduction and clustering algorithms. A representative example is Latent Dirichlet Allocation (LDA) (**?**). LDA is a generative model that enables automatic extraction of topics in a corpus, and which topics are covered in each document. However, generation of documents using LDA leads to utterly incomprehensible text. GANs belong to the former category. Non-parametric models (e.g., histograms and kernel density estimation) suffer from slow converge even in the simplest low dimensional problems (**???**). In contrast, GANs seems to be immune to this issue. For instance, GANs were able to generate compelling images from very limited data (**???**).

Unfortunately, current work on GANs overlooks very important issues regarding the evaluation of their ability to generate samples from the correct distribution. We believe that a meaningful evaluation should *quantify* how close the output of a GAN is to the underlying distribution. This is in contrast with current visual tests typical of recent GAN work, which we consider to be flawed. For instance, from afar a 32-by-32 pixel images from (**?**) might look like a bedroom. However, as soon as each image is magnified and examined individually, they start to appear somewhat *off*. This is even more evident in higher resolution images, which look even more unnatural (**?**). (The same phenomenon is evident in text generation (**?**).) Proper, principled quantification of the quality of GANs' output would enable us to determine a specific set of applications in which GANs can be instrumental.

Most of the current results on GANs focus on addressing the alleged instability of GANs using ad-hoc approaches that make training easier and more effective (**?????????????**)[1]. As a result, these paper claim that they are able to generate better-looking images, more realistic text, etc. However, very few papers have *measured* how well GAN correctly generate samples from the underlying distribution; instead, they simply show indirect evidence that GANs are well suited for approximating these distributions. In a way, the research community is exploring *what we can do to improve GANs*, rather than asking *what are GANs useful for?*

We hypothesize that GANs are unable to deliver accurate density estimates in high-dimensional structured spaces. On the other hand, there is plenty of evidence that GANs are capturing some relevant information from the data. This is important, because it might enable the use of GANs as universal simulators even though density estimates are wrong. There are some applications (**????**) in which this is clearly the case. For example, accurate density estimation is not required for GAN-based password generation (**?**), as long as the GAN is able to capture passwords properties that

[1]This is a very active area of research, and as such we do not intend this list to be comprehensive.

previous methods could not model. While frequencies might be off, the elements generated by the GAN in (**?**) are valid passwords nonetheless.

GAN-like objective functions have also emerged in the context of approximate variational inference with implicit distributions (**??**). Many applications leverage this kind of highly-scalable approximate inference machinery to train interpretable latent variable models in which an accurate estimate of the underlying distribution is not mandatory (see, e.g., (**??**)). However, it is not clear whether GANs can be safely used in an application that requires some form of density estimation. We expect that, ultimately, this is application dependent, as it can take the form of a flawed density estimate or rely on some statistics of the models that match those in the real data or described a interpretable model.

In this paper, to foster the discussion about where GANs should lead us, we first examine the convergence proofs for GANs, and why we believe they are incomplete or not suitable for analyzing GANs. We also revisit the papers that evaluated GANs density estimate, and why most of them are critical of GANs. In the second part of this paper, we show two examples that reinforce this idea that GANs density estimates are very poor, and that therefore GANs do not generate from the same distribution the data is coming from. Our experiments concur with those presented in the papers detailed in the second part of the paper.

2 Density Estimation for GANs

2.1 Proofs of convergence

At the time of this writing, there are four papers that analyze the convergence of the GANs to the true density. The first proof that GANs can converge to the true density was given in the original GAN paper (**?**). Recently, a new convergence proof has been reported in (**??**). Arora et al. reported that there exist equilibrium points during the training of the GAN for which the generative neural network distribution would not converge to the real distribution of the data (**?**).

The analysis in the original GAN paper (**?**) proved that, if we knew the underlying density model, the equilibrium of the game would be a generator that draws samples from the original distribution and a discriminator that cannot distinguish between the output of the generator and the original distribution. This assumed that if the discriminator has finite (VC) dimension, and the number of samples tends to infinity, then the model will converge. Unfortunately, however, there are several missing steps in the derivation. First, the law of large numbers is not enough for that convergence to take place, since one will need to prove uniform convergence too (**?**). Second, we need to prove that the iterative procedure between the discriminator and the generator does not get stuck in local minima in which the generator only mimics, at best, part of the distribution. Third, we also need to analyze the convergence when the neural network grows, and so does the input dimension (e.g., noise distribution). Failure to address these three issues might lead to dropping modes or not estimating the tails of the distribution correctly. This problem has not even been addressed yet, as we show in the experimental section. Finally, there is no bound for finite set of samples, so we will never know how far we are from fully modeling the input distribution.

In (**?**), the authors propose a GAN that follows a boosting procedure in which new components are added to the mixture until the original distribution is recovered. The authors show that this technique leads to exponential convergence to the underlying density. Our concern with the proof is its practical applicability: at each step, the proof requires that the GAN estimated density dQ and the true underlying density of the data dP_d satisfy $\beta dQ \leq dP_d$. However, it is not clear how to design a generative network that induces a density dQ that would guarantee $\beta dQ \leq dP_d$ with a non-zero β when dP_d is a high-dimensional structure generative process.

In (**?**) the authors prove that for standard metrics (e.g. Shannon-Jensen divergence and Wasserstein-like integral probability metrics), the discriminator might stop discriminating before the estimated distribution converges to the density of the data. They also show a weaker result, in which convergence might happen but the estimated density might be off, when the discriminator based on a deep neural network is not large enough (sufficient VC dimension).

Finally, in (**?**) the authors focus on two important aspects of GAN convergence: (1) how well the generative distribution approximates the real distribution, and (2) when this convergence takes place. With respect to (1), they show that the discriminator forces some kind of moment matching between

the real distribution and the output of the generator. In order to get full representation of the density, the discriminator must grow with the data. Regarding (2), they show a week convergence result. This result is somewhat complementary to **?**, because it indicates that the discriminator complexity needs to grow indefinitely to achieve convergence. The question that remains to be answer is the rate of convergence, as the moments that need to be matched for complicated distributions might require large data and complex discriminators. As a result, in practice we cannot tell if the generated distribution is close enough to the distribution we are interested in.

Today's GAN's results can be explain in the light of these theoretical results. First, we are clearly matching some moments that relate to visual quality of natural images with finite size deep neural networks. However, these networks might not be deep enough to capture all relevant moments, and hence they will not be able to match the actual distribution of these images. This explains why low resolution images, with which the interpolation performed by the human visual systems plays an important role **?**, look very realistic, while large resolution the results are far from acceptable.

2.2 CONVERGENCE CRITIQUES

In two recent papers (**??**), the authors proposed to use two-sample tests to validate GANs density estimates. These methods notice that there are artifacts in which the two-sample tests can lock onto to distinguish between real and fake images. As a result, both papers reach similar conclusions: if we compare images in pixel space, no distribution generated by the GAN passes those tests. In the next section we replicate these experiments and find similar conclusions.

In (**?**) the authors replicated these experiment with the MNIST dataset in **?**. In these experiments, humans could not distinguish between fake and real digits. The authors showed that, for the MMD test, distinguishing was easy because of the artifacts and because the GANs were only showing standard digits, and were unable to generate samples from the tail of the distribution. This experiment shows the downsides of using human evaluation as a proxy for densities estimates, as humans focus more on normalcy and under-represent the unexpected. In the next section, we show another test that illustrates that GANs might be only focusing on the modes in the data and ignoring those tails, which might be critical in applications in which the tails carry the information of the extreme events that we might be interested in (e.g., rare genetic diseases, or extreme weather patterns).

In (**?**), the authors also showed that if instead of comparing in the pixel space, the comparison is made in some transformed space (in (**?**), the transformed space was the final layer in a Resnet structure), the fake samples and the true samples were indistinguishable. This result is quite appealing, because there are some statistics in which the GANs samples are distributed like the real samples. If those statistics are sufficient for the problem at hand, then we might be able to rely on GANs as simulators. It is therefore important to know for which statistics this happens, and how broad these problems are.

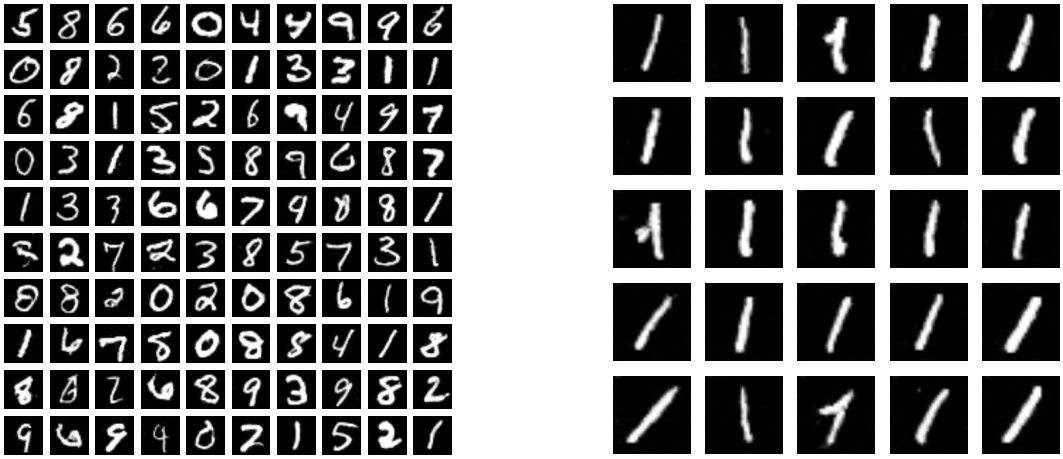
In (**??**), the authors complete their theoretical work (**?**) to show that GANs might be dropping modes of the distribution by adapting the birthday paradox for images. Since images comes from a continuous space, they need some discretization in order to show this mode dropping effect. In this paper, we show that this limitation can be also analyzed when the GAN is used to generate a discrete distribution. We show that the estimated probabilities are way off, and we find both mode dropping and mode enhancing to happen at the same time.

3 SIMULATIONS

We now provide a set of experiments to support our previous discussion. We first consider two image datasets, MNIST and CIFAR-10, and later we revisit the recently proposed GAN-approach to enhance password guessing (PassGAN) (**?**). While in the first case we have used the GAN implementation proposed in (**?**), PassGAN has been implemented using the improved Wasserstein GAN described in (**?**).

3.1 SIMULATIONS USING MNIST AND CIFAR DATABASES

We consider a specific implementation of the generator and discriminator networks, both trained over the MNIST (60,000 labeled images of handwritten numbers, plus 10,000 extra images in the



(a) (b)

Figure 1: Samples drawn from the generator network trained over the MNIST and 1-MNIST datasets.

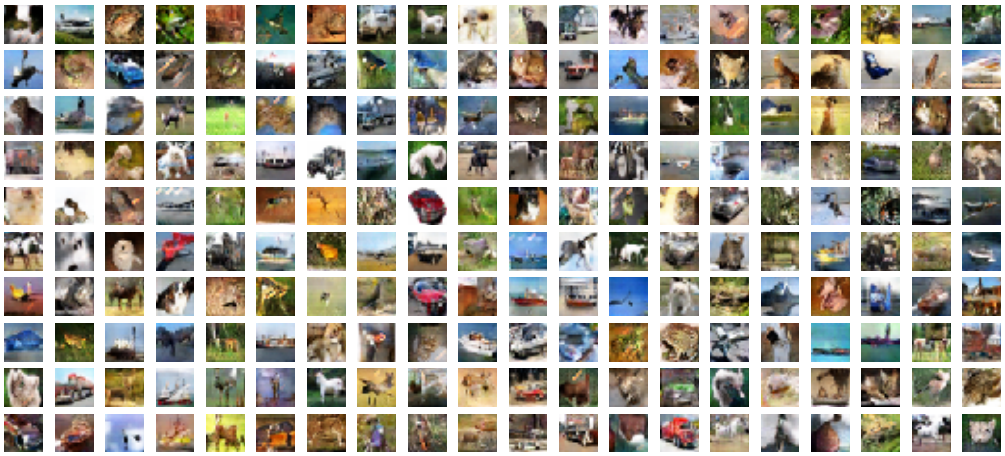


Figure 2: 200 samples drawn from the generator network trained over CIFAR-10 dataset.

test set) and CIFAR-10 datasets (50,000 labeled natural images, plus 10,000 extra images in the test set). Also, we train the GAN with the *one*-s from MNIST, i.e., the 1-MNIST database, which contains 6739 training images and 1134 test images.

We have reproduced the experiments and network design of (**?**). More specifically, the generator network is a 4 layer deep CNN, with both batch and weight normalization (**?**). The input $\mathbf{z}$ is a 100-dimensional Gaussian. The discriminator network is a 9 layer deep convolutional network with dropout, weight normalization and a Gaussian noise layer that is added to the input. The discriminator is trained using minibatch discrimination (MD), which helps to avoid collapse of the generator by looking at multiple data examples in combination. As described in (**?**), within a few iterations of stochastic gradient descend (SGD), the generator network is already able to provide visually ap-

pealing images. In all cases, we have run 600 epochs, where per iteration all data mini-batches are processed with SGD once. In Figure **??**, we illustrate samples from the generator network trained over MNIST (a) and 1-MNIST (b), and in Figure **??** samples when it is trained over CIFAR-10. In most cases the samples are visually appealing, and we expect that a person would be unable to distinguish between real images, and images generated by the neural network. (These results match those in **?**.)

In the next section we analyze the Kernel Two-Sample Test results. Results that show mode dropping and under sampling of the tails of the distribution are shown in Appendix.

3.1.1 KERNEL TWO-SAMPLE TEST

In contrast with (**?**), where the inception score is introduced as a metric of the quality of the images sampled from the generator network, we propose a statistical hypothesis test to study whether the distributions $p(\mathbf{x})$ and the probability distribution of images induced by the generator network, i.e., $p_g(\mathbf{x}) = G(\mathbf{z}_i; \theta_G)$, are different. This problem is known as the Two-Sample Test (**?**). We use the Kernel Two-Sample Test proposed in (**?**), where the test statistic is the Maximum Mean Discrepancy (MMD), defined as the largest difference in expectations over functions in a unit ball of a characteristic reproducing kernel Hilbert space (RKHS). When MMD is large, the samples are likely from different distributions. In (**?**), unbiased empirical estimators to the MMD metric are presented, and further, it is shown that the empirical MMD estimate shows a statistically significant difference between distributions. Given i.i.d. samples $X \sim p(\mathbf{x})$ and $\tilde{X} \sim p_g(\mathbf{x})$, statistical hypothesis testing is used to distinguish between the null hypothesis $\mathcal{H}_0 : p = p_g$ and the alternative hypothesis $\mathcal{H}_a : p \neq p_g$. This is achieved by comparing the empirical MMD estimate with a particular threshold ϵ: if the threshold is exceeded, then the test rejects the null hypothesis.

Given the *level* α of the test (i.e., an upper bound on the Type I error), we use bootstrap resampling on the aggregated data to obtain a test threshold ϵ.[2] In our experiments we have used 1000 bootstrap shuffles, the radial basis function kernel with a bandwidth set as median distance between points in the aggregate sample. Also, we fix $\alpha = 0.05$. We use N_{Test} samples taken at random from the MNIST and CIFAR-10 *test* sets and the same number of samples drawn from the generator network. Results are averaged over 1000 realizations. In Table **??** (a) we include the percentage of times the null hypothesis $\mathcal{H}_0 : p = p_g$ was rejected. We also include in the table the average MMD value and two times its standard deviation, as well as the estimated threshold ϵ. In Table **??** (b), we randomly split the MNIST and CIFAR-10 test datasets into two disjoint sets, and we perform the same statistical test, where ideally the null hypothesis $\mathcal{H}_0$ should never be rejected. The comparison of results in Table **??** (a) and (b) clearly shows that as the number of data points increase, the empirical MMD gets further from the test threshold, strongly indicating a significant statistical difference between distributions. In contrast, the MMD metric computed in Table **??** (b) does not grow with N_{Test} and remains very close to the threshold ϵ. As a result, we can safely conclude that the GAN density model does not match the distribution of the training images. As mentioned before, the use of Two-Sample statistical hypothesis test to study the distribution induced by GAN has been also proposed in (**??**) with similar conclusions.

3.2 PASSGAN, A GENERATIVE MODEL FOR PASSWORD GUESSING

PassGAN is a recent technique proposed in (**?**), which trains a character-level GAN on leaked password datasets. It aims at learning from the underlying training data distribution (*i.e. the distribution of leaked passwords*), autonomously understanding the characteristics that constitute a password. PassGAN builds upon the work of **?**, which shows impressive results on GAN-based text generation. We have reproduced the results of PassGAN to demonstrate severe mode-dropping, which further supports the conclusions from previous experiments. We also illustrate how the distribution induced by PassGAN presents modes that do not exits in the real distribution of passwords. While this shows that the density estimate is not correct, these new modes are not completely random: specifically, these outputs are *plausible* human-generated passwords.

After training PassGAN, we used sample from generative network to obtain "fresh" password samples which were then used for password guessing. Using the same network design as in **?**, we

[2] See `http://www.gatsby.ucl.ac.uk/~gretton/mmd/mmd.htm`

Table 1: In (a), we use the Kernel Two-Sample Test with real images and samples drawn from the generator network. In (b), we use instead real images taken from two disjoint sets on the MNIST and CIFAR-10 test datasets.

Data Set	N_{Test}	% $\mathcal{H}_0$ rejected	MMD	ϵ
MNIST	100	84	1.41 ± 0.46	1.16
MNIST	1000	100	2.80 ± 0.77	1.24
MNIST	5000	100	8.96 ± 1.45	1.24
CIFAR-10	100	95	1.76 ± 1.12	1.16
CIFAR-10	1000	100	5.64 ± 2.66	1.13
CIFAR-10	5000	100	45.15 ± 7.20	1.11

(a)

Data Set	N_{Test}	% $\mathcal{H}_0$ rejected	MMD	ϵ
MNIST	100	59	1.24 ± 0.36	1.18
MNIST	1000	44	1.25 ± 0.41	1.24
MNIST	5000	46	1.27 ± 0.34	1.24
CIFAR-10	100	46	1.24 ± 0.43	1.18
CIFAR-10	1000	58	1.35 ± 0.76	1.21
CIFAR-10	5000	61	1.34 ± 0.72	1.19

(b)

trained a Wasserstein-GAN on a leaked portion of RockYou dataset (**?**) using 23,679,744 (80%) passwords for training (9,925,896 unique entries), and the remaining 5,919,936 passwords as testing data (3,094,199 unique entries). The results achieved from such process were competitive with state-of-the-art rule-based approaches such as in John-the-Ripper (**?**) or Hashcat (**?**) password cracking tools. (See also (**?**) for further details.)

We generated 10^{10} samples with PassGAN, out of which 528,834,530 (52.88%) were unique samples. These samples matched 2,774,269 passwords (46.9%) from the testing set. When checked on a separate (i.e., unknown to PassGAN) passwords dataset (**?**), PassGAN was able to guess 4,996,980 (11.5%) passwords out of 43,354,871 unique entries.

Table **??** shows the top 30 most frequent passwords in the RockYou dataset, sorted in decreasing order, used during PassGAN training. The second column provides the number of occurrences of a password in the dataset, whereas the third column provides the frequency of such a password in the training data. In the fourth column, we present the GAN estimated frequency which represents the number of times a certain password was generated with respect to the total number of generations. The generator does not follow the underlying distribution, as there are passwords that are clearly underrepresented by the GAN (or even mode dropping). (See, e.g., "password", "abc123", "babygirl", and "anthony". Passwords present in the training data, but not generated by PassGAN, are shown in Table **??** as "not generated". The results in this table show a severe mode-dropping effect, because the 30th most common password with over 8,000 repetition in the over 23 million passwords is not generated by PassGAN within its first 10^{10} outputs.

In Table **??** we show the 30 passwords most frequently produced by PassGAN. Interestingly, the model produces also passwords that have low or no representation in the training dataset, such as "iluv!u&", which is not present in the training data, and "123256", which appears only 6-times in the training set. PassGAN generated new modes that are somewhat structured, and are not simply sequences of random letters: passwords such as "dangel", "michel", or "ilove" might in fact be real passwords.

Table 2: Top-30 most frequent passwords present on the dataset used to train PassGAN. The first column shows the password; the second column the number of occurrences of the password in the training data; the third column, the frequency in the training data; and in the forth column, we provide the GAN estimated frequency.

Password	**Occurrence in Training Data**	**Frequency in Training Data**	**GAN Estimated Frequency**
123456	232,844	0.98%	100,971,288 (1.0%)
12345	63,135	0.27%	21,614,548 (0.22%)
123456789	61,531	0.26%	22,208,040 (0.22%)
password	47,507	0.20%	85,889 (8.6e-4%)
iloveyou	40,037	0.17%	10,056,700 (0.10%)
princess	26,669	0.11%	190,796 (0.0019%)
1234567	17,399	0.073%	7,545,708 (0.075%)
rockyou	16,765	0.071%	55,515 (5.5e-4%)
12345678	16,536	0.070%	5,070,673 (0.051%)
abc123	13,243	0.056%	6,545 (6.5e-5%)
nicole	12,992	0.055%	206,277 (0.0021%)
daniel	12,337	0.052%	3,304,567 (0.033%)
babygirl	12,130	0.051%	13,076 (1.3e-4%)
monkey	11,726	0.050%	116,602 (0.0012%)
lovely	11,533	0.049%	1,026,362 (0.010%)
jessica	11,262	0.048%	220,849 (0.0022%)
654321	11,181	0.047%	19,912 (1.9e-4%)
michael	11,174	0.047%	517 (5.2e-6%)
ashley	10,741	0.045%	116,858 (0.0012%)
qwerty	10,730	0.045%	135,124 (0.0013%)
iloveu	10,587	0.045%	4,839,368 (0.048%)
111111	10,529	0.044%	101,903 (0.0010%)
000000	10,412	0.044%	108,300 (0.0011%)
michelle	10,210	0.043%	739,220 (0.0073%)
tigger	9,381	0.040%	658,360 (0.0066%)
sunshine	9,252	0.039%	3,628 (3.6e-5%)
chocolate	9,012	0.038%	12 (1.2e-7%)
password1	8,916	0.038%	6,427 (6.4e-5%)
soccer	8,752	0.037%	25 (2.5e-7%)
anthony	8,752	0.036%	not generated

ACKNOWLEDGMENT

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X Pascal GPU used for this research.

REFERENCES

N.H. Anderson, P. Hall, and D.M. Titterington. Two-sample test statistics for measuring discrepancies between two multivariate probability density functions using kernel-based density estimates. *Journal of Multivariate Analysis*, 50(1):41 – 54, 1994.

Martin Arjovsky and Leon Bottou. Towards principled methods for training generative adversarial networks. *5th International Conference on Learning Representations (ICLR)*, 2017.

Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *CoRR*, abs/1701.07875, 2017. URL `http://arxiv.org/abs/1701.07875`.

Sanjeev Arora and Yi Zhang. Do gans actually learn the distribution? an empirical study. *ICLR*, 2018. URL `http://arxiv.org/abs/1706.08224`.

Table 3: 30 most frequent passwords produced by PassGAN after generating 10^{10} Samples. The first column shows the password sample; the second column, the number of occurrences; and the third column shows the frequency of the password.

GAN generated Passwords	**Occurrence in GAN**	**Frequency in GAN**	**Frequency in Training Data**
123456	100,971,288	1.01%	232,844 (0.98%)
123456789	22,208,040	0.22%	61,531 (0.26%)
12345	21,614,548	0.22%	63,135 (0.27%)
iloveyou	10,056,700	0.10%	40,037 (0.17%)
1234567	7,545,708	0.075%	17,399 (0.073%)
angel	6,384,511	0.064%	8,425 (0.036%)
12345678	5,070,673	0.051%	16,536 (0.070%)
iloveu	4,839,368	0.048%	10,587 (0.045%)
angela	3,377,148	0.034%	4,548 (0.019%)
daniel	3,304,567	0.033%	12,337 (0.052%)
sweety	2,560,589	0.026%	5,140 (0.022%)
angels	2,455,602	0.025%	6,600 (0.028%)
maria	1,582,718	0.016%	3,178 (0.013%)
loveyou	1,541,431	0.015%	6,797 (0.029%)
andrew	1,307,400	0.013%	6,666 (0.028%)
123256	1,294,044	0.013%	6 (2.5e-5%)
iluv!u	1,268,315	0.013%	0 (0%)
dangel	1,233,188	0.012%	43 (1.8e-4%)
michel	1,190,127	0.012%	794 (0.0033%)
marie	1,187,051	0.012%	1,788 (0.0076%)
andres	1,055,809	0.011%	3,016 (0.013%)
lovely	1,026,362	0.010%	11,533 (0.049%)
123458	989,324	0.010%	181 (7.6e-4%)
sweet	968,822	0.010%	2,366 (0.010%)
prince	920,415	0.0092%	2,883 (0.012%)
ilove	888,109	0.0089%	555 (0.0023%)
hello	861,067	0.0086%	6,270 (0.026%)
angel1	840,056	0.0084%	3,454 (0.015%)
iluveu	826,944	0.0083%	30 (1.3e-4%)
723456	820,268	0.0082%	6 (2.5e-5%)

Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. Generalization and equilibrium in generative adversarial nets (gans). In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pp. 224–232, 2017. URL http://proceedings.mlr.press/v70/arora17a.html.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003. ISSN 1532-4435.

M Fahle and T Poggio. Visual hyperacuity: spatiotemporal interpolation in human vision. *Proc. R. Soc. Lond. B*, 213(1193):451–477, 1981.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 27*, pp. 2672–2680. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13:723–773, March 2012.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans, 2017. URL http://arxiv.org/abs/1704.00028.

HashCat, 2017. URL https://hashcat.net.

Karol Hausman, Yevgen Chebotar, Stefan Schaal, Gaurav S. Sukhatme, and Joseph Lim. Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets. *CoRR*, abs/1705.10479, 2017. URL http://arxiv.org/abs/1705.10479.

Jamie Hayes and George Danezis. ste-gan-ography: Generating steganographic images via adversarial training. *CoRR*, abs/1703.00371, 2017. URL http://arxiv.org/abs/1703.00371.

Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the GAN: Information leakage from collaborative deep learning. *CCS'17*, 2017a. URL https://arxiv.org/pdf/1702.07464.pdf.

Briland Hitaj, Paolo Gasti, Giuseppe Ateniese, and Fernando Perez-Cruz. PassGAN: A Deep Learning Approach for Password Guessing. *CoRR*, abs/1709.00440, 2017b. URL https://arxiv.org/pdf/1709.00440.pdf.

Huszár, Ferenc. Variational inference using implicit distributions. *CoRR*, 2017. URL https://arxiv.org/pdf/1702.08235.pdf.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In David Blei and Francis Bach (eds.), *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 448–456. JMLR Workshop and Conference Proceedings, 2015. URL http://jmlr.org/proceedings/papers/v37/ioffe15.pdf.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*. 2014.

Holden Lee, Rong Ge, Tengyu Ma, Andrej Risteski, and Sanjeev Arora. On the ability of neural nets to express distributions. In *Proceedings of the 30th Conference on Learning Theory, COLT 2017, Amsterdam, The Netherlands, 7-10 July 2017*, pp. 1271–1296, 2017. URL http://proceedings.mlr.press/v65/lee17a.html.

Yujia Li, Kevin Swersky, and Richard S. Zemel. Generative moment matching networks. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pp. 1718–1727, 2015. URL http://jmlr.org/proceedings/papers/v37/li15.html.

LinkedIn. Linkedin. URL https://hashes.org/public.php.

Shuang Liu, Olivier Bousquet, and Kamalika Chaudhuri. Approximation and convergence properties of generative adversarial learning. *CoRR*, abs/1705.08991, 2017. URL http://arxiv.org/abs/1705.08991.

David Lopez-Paz and Maxime Oquab. Revisiting Classifier Two-Sample Tests for GAN Evaluation and Causal Discovery. *5th International Conference on Learning Representations (ICLR)*, 2017. URL http://arxiv.org/abs/1610.06545.

Gabor Lugosi and Andrew Nobel. Consistency of data-driven histogram methods for density estimation and classification. *Ann. Statist.*, 24(2):687–706, 1996.

Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. On distinguishability criteria for estimating generative models. *34th International Conference on Machine Learning (ICML 2017)*, 2017. URL https://arxiv.org/pdf/1701.04722.pdf.

Lars M. Mescheder, Sebastian Nowozin, and Andreas Geiger. The numerics of gans. *CoRR*, abs/1705.10461, 2017. URL http://arxiv.org/abs/1705.10461.

Shakir Mohamed and Balaji Lakshminarayanan. Learning in implicit generative models. *5th International Conference on Learning Representations (ICLR)*, 2017.

Youssef Mroueh and Tom Sercu. Fisher GAN. *CoRR*, abs/1705.09675, 2017. URL http://arxiv.org/abs/1705.09675.

Youssef Mroueh, Tom Sercu, and Vaibhava Goel. Mcgan: Mean and covariance feature matching GAN. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pp. 2527–2535, 2017. URL http://proceedings.mlr.press/v70/mroueh17a.html.

Vaishnavh Nagarajan and J. Zico Kolter. Gradient descent GAN optimization is locally stable. *CoRR*, abs/1706.04156, 2017. URL http://arxiv.org/abs/1706.04156.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. pp. 271–279, 2016.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *4th International Conference on Learning Representations (ICLR)*, 2016.

RockYou. Rockyou, 2010. URL http://downloads.skullsecurity.org/passwords/rockyou.txt.bz2.

Kevin Roth, Aurélien Lucchi, Sebastian Nowozin, and Thomas Hofmann. Stabilizing training of generative adversarial networks through regularization. *CoRR*, abs/1705.09367, 2017. URL http://arxiv.org/abs/1705.09367.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training gans. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 29*, pp. 2234–2242. Curran Associates, Inc., 2016. URL http://papers.nips.cc/paper/6125-improved-techniques-for-training-gans.pdf.

B. Schölkopf and A. Smola. *Learning with kernels*. M.I.T. Press, 2002.

Akash Srivastava and Charles Sutton. Autoencoding variational inference for topic models. *International Conference on Learning Representations (ICLR)*, 2017. URL https://arxiv.org/pdf/1703.01488.pdf.

Dougal J. Sutherland, Hsiao-Yu Tung, Heiko Strathmann, Soumyajit De, Aaditya Ramdas, Alex Smola, and Arthur Gretton. Generative models and model criticism via optimized maximum mean discrepancy. *International Conference on Learning Representations (ICLR)*, 2017. URL https://arxiv.org/pdf/1703.01488.pdf.

John the Ripper, 2017. URL http://www.openwall.com/john/.

Ilya O. Tolstikhin, Sylvain Gelly, Olivier Bousquet, Carl-Johann Simon-Gabriel, and Bernhard Schölkopf. Adagan: Boosting generative models. *CoRR*, abs/1701.02386, 2017. URL http://arxiv.org/abs/1701.02386.

Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

Larry Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer Publishing Company, Incorporated, 2010. ISBN 1441923225, 9781441923226.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. Improved variational autoencoders for text modeling using dilated convolutions. *34th International Conference on Machine Learning (ICML 2017)*, 2017. URL http://proceedings.mlr.press/v70/yang17d/yang17d.pdf.

Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. Adversarial feature matching for text generation. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pp. 4006–4015, 2017. URL http://proceedings.mlr.press/v70/zhang17b.html.

APPENDIX: EXPLAINING THE GAP: NEAREST NEIGHBOR ANALYSIS BETWEEN SAMPLE SETS

The samples generated by the GAN failed the MMD two sample test even though the generated images look as good as the original images in MNIST and CIFAR-10. We believe that the GAN fails the test because the samples are too concentrated on the modes and does not explore the full density model. This is why the samples generated for MNIST all look very good and we are almost never surprised by the weird digits that we find sometimes in the original MNIST.

We now perform a simple experiment to demonstrate the generative distribution overfit to the dominant modes of $p(\mathbf{x})$. Consider first the 1-MNIST dataset, with 6739 training images. After jointly training the discriminative and generative networks, we sample from the generator network a set of 6739 images and, for each of them, we compute the nearest neighbor (NN) in the 1-MNIST training set. We denote by $n_{\text{1-MNIST}\to\text{GAN}}(i)$ the number of times that the i-th image in the 1-MNIST training set is the NN of images in the fake dataset generated by the GAN. Note that large $n_{\text{1-MNIST}\to\text{GAN}}(i)$ values will correspond to images from the 1-MNIST that the GAN tends to over reproduce. The $n_{\text{1-MNIST}\to\text{GAN}}(i)$ profile is shown in Figure **??**, where we have sorted the index images according to increasing values of $n_{\text{1-MNIST}\to\text{GAN}}(i)$.

We compare this result with the leave-one-out (LOO) NN profile for the 6739 images used for training, denoted by $n_{\text{1-MNIST}\to\text{1-MNIST}}(i)$. This profile is shown by a red dashed line in Figure **??**. The results clearly indicate that there is a subset in the 1-MNIST training set that is over-reproduced in the generated artificial dataset, and thus, they correspond to modes in $p_g(\mathbf{x})$ that are not so-dominant in $p(\mathbf{x})$. On the other end of the spectrum the GAN represents much fewer images than the LOO NN profile (almost 1000 samples less). This profile leads us to believe that the GAN density estimation over represents the modes and under represents the tails of the distribution.

In Figure **??** the $n_{\text{1-MNIST}\to\text{GAN}}(i)$ profile is computed for 10^5 generated images *and the two NNs are found for each image*. In this case, there is still 1,300 images of ones (20% of the 1-MNIST database) that do not have a first or second NNs in 100,000 GAN generated images. The image that is most popular appears as the first or second NN in 0.4% of the cases, which seems to corroborate our initial findings that the GAN over samples the modes and under sample the tails of $p(\mathbf{x})$.

An alternative illustration that reinforces the idea that the distribution $p_g(\mathbf{x})$ is indeed not representing the tails of $p(\mathbf{x})$ is provided in Figure **??**, where in column (a) we represent the two ones from 1-MNIST training set furthest from their LOO NNs. In (b), we represent the two images in the fake dataset sampled from the generative network (using 10^5 images) that are furthest from their LOO NNs. While in the ones from the real database, Figure **??**(a), we observe images with marked calligraphic style (a reasonable *unlikely* one), in (b) we can appreciate images where the large distance is explained by residual noise rather than by an unusual calligraphic style. The first column of images will never be reproduced by the GAN, while the second set are noisy regular-one images.

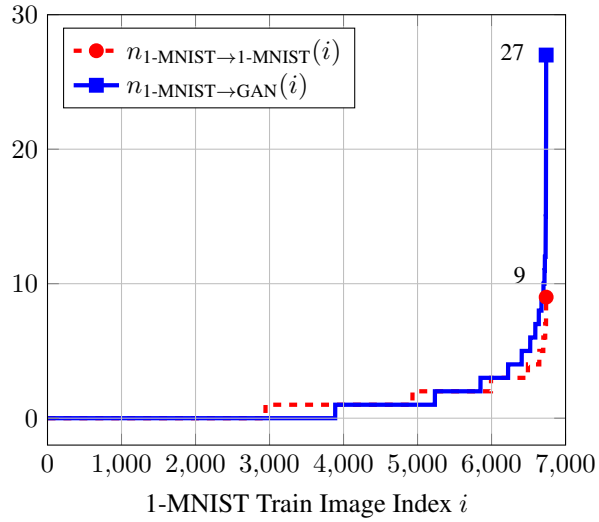


Figure 3: We represent the $n_{\text{1-MNIST}\to\text{GAN}}(i)$ and $n_{\text{1-MNIST}\to\text{1-MNIST}}(i)$ profiles computed for 6739 images sampled at random from the generator network and for the 6739 images in the 1-MNIST training set.

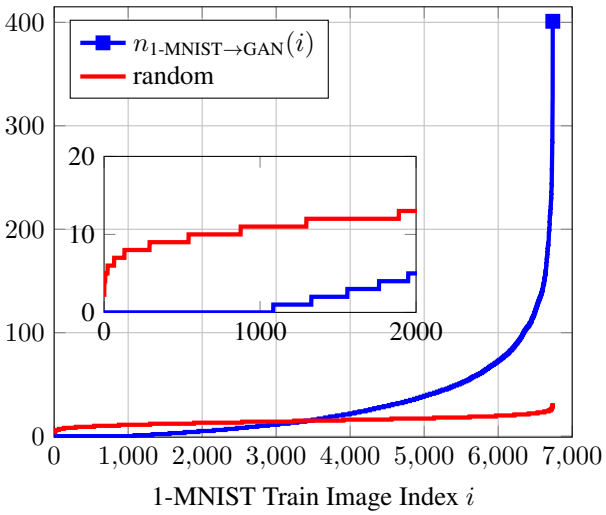
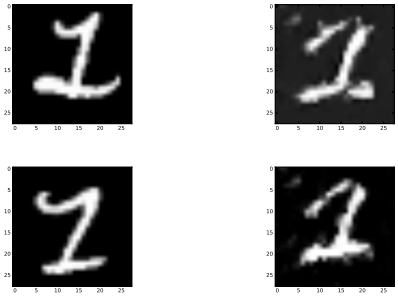


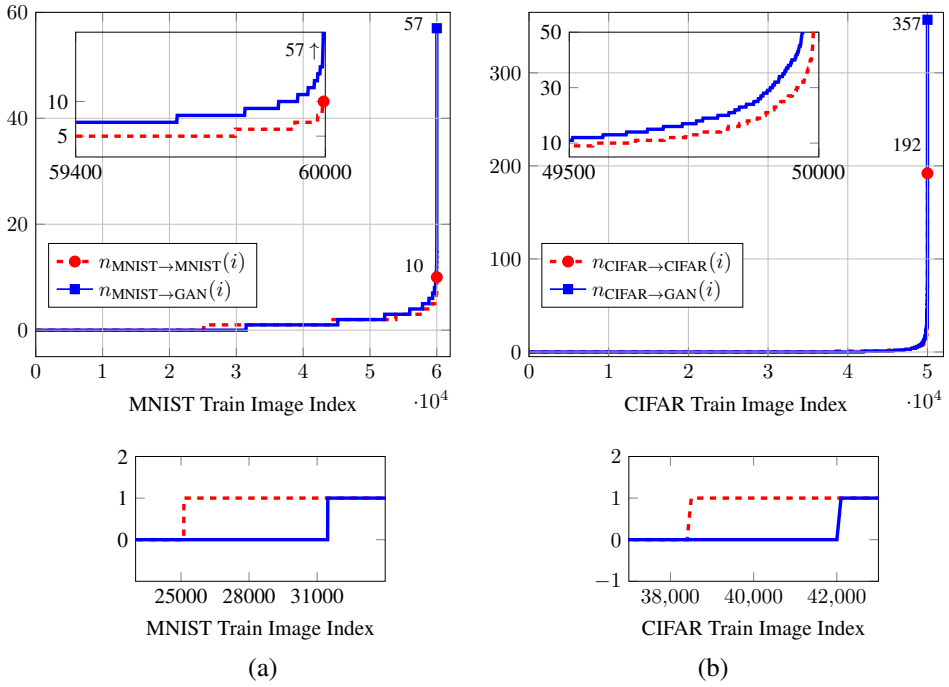
Figure 4: We represent the $n_{\text{1-MNIST}\to\text{GAN}}(i)$ profile computed for 10^5 images sampled at random from the generator network using the two NNs of each image.

In Figures **??** we have reproduced the same experiments for the full MNIST train set (a), where we have sample at random 60,000 images for the GAN, and the CIFAR-10 training set (b), where 50,000 images have been sampled from the GAN. Results in both cases corroborate our conclusions, showing that the distribution $p_g(\mathbf{x})$ overrepresents certain subsets of images of the training set and under represent the tails of the distribution. For the NMIST data the 1% most popular images represent 8.32% of the images in the GAN dataset and only 5.48% in the original MNIST data, while needing 6000 less images to find the NNs for all GAN generated images. This shows that the GAN covers a significantly less fraction of the original distribution and overfits to the modes. For the CIFAR-10, the results are even more extreme.



(b) (c)

Figure 5: In (b), we represent the two images that are furthest from their neighbors within the MNIST training set. In (b), we represent the two images that are furthest from their neighbors in artificial set of images sampled from the generative network.



(a) (b)

Figure 6: In (a), we represent $n_{\text{MNIST}\to\text{GAN}}(i)$ and $n_{\text{MNIST}\to\text{MNIST}}(i)$ computed for MNIST with 60,000 images. In (b), we reproduce the experiments for CIFAR with 50,000 images.