An Empirical Study of the Generalization Behavior of Generative Adversarial Networks

Hongyu Ren
Peking University
rhy@pku.edu.cn

Shengjia Zhao
Stanford University
sjzhao@cs.stanford.edu

Jiaming Song
Stanford University
tsong@cs.stanford.edu

Lijie Fan
Tsinghua University
flj14@mails.tsinghua.edu.cn

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

Abstract

We conduct an empirical evaluation of the performance of generative adversarial networks (GANs) by comparing high level properties of the samples, such as the distribution of generated labels. We propose to measure the complexity of the label distribution through the minimum description length under a certain encoding scheme. Preliminary empirical results suggest that it is more difficult for GANs to learn the distribution when its label distribution requires longer codes. In particular, we discover that GANs can memorize facts better than exceptions, and that they do not necessarily perform worse when there are more modes in the true distribution.

1 Introduction

Generative Adversarial Networks (GANs), [1], perform well on a range of tasks including image generation [2], style transfer [3], semi-supervised learning [4, 5], and imitation learning [6, 7]. One important application of generative models is to learn disentangled latent factors of variation, and generalize to novel combinations. Statistical learning theory [8] provides guaranteed generalization bounds for supervised learning, but is difficult to apply to generative modeling tasks. In fact even defining the notion of "correct generalization" is difficult. For example, upon learning on a dataset with red and green squares, and red triangles, will/should the model generate green triangles?

Answering this question requires human intuition and judgment. A typical example is to traverse the manifold [2] by constructing a path in the latent feature space from one example to another, and observe how the output varies. A model that generalizes should generate a meaningful sequence of interpolations between the two examples. In addition, each step of the interpolation should agree with human judgment. For example, the existence of white and black flowers and white snow does not imply the existence of black snow, so such an interpolation would be considered "incorrect" by human experience.

This paper attempts to explore this question. We design a synthetic dataset to quantitatively evaluate the generalization behavior of GANs by combining existing real world datasets, e.g., generating images with multiple MNIST digits. We train a generative model on a subset of all possible three digit combinations, and observe when GANs generalize to unseen combinations. We observe that the generalization behavior can be explained with the principle of Minimum Description Length [9]. When a generative model can effectively memorize all valid combinations, no generalization will/should occur. Generalization occurs only when a generative model can no longer remember all valid combinations, and it will select an approximating set of label combinations that is easier to represent.

We also observe empirically that GANs have a surprisingly small memorization capacity, and start to generalize when given only a small number of label combinations during training (as opposed to

Figure 1: Illustration for the experiment in Section 2.

just 'memorizing' these cases). This is surprising since from the observations in [10], a feed-forward neural network can memorize a huge number of random label combinations.

2 Learning to Generate Random Labels

We consider a family of distributions where each sample from the distribution is a 56×56 image that contains three MNIST digits at fixed locations in the image (top left, top right, bottom left, 28×28 each). Hence, there are a total of $10^3 = 1000$ *combinations*, e.g., '7,1,7' or '9,1,3'.

Training Data Generation: We select randomly a subset of these 1000 combinations, specifically, 1%, 4%, 16%, 32%, 50%, 84%, 96%, 99% of the combinations. We call these sets of combinations $C_1, C_4, \cdots, C_{999} \subseteq \{0, \cdots, 9\}^3$. We define p as the "true combination distribution", which assigns equal probability to the selected combinations. For example, if we select two combinations $(7, 1, 7)$ and $(9, 1, 3)$, then $p(7, 1, 7) = p(9, 1, 3) = 0.5$. For each C_i, we construct a training dataset of images with three digits each, only allowing digit combinations that are in C_i, combining individual images of digits from MNIST. We denote the datasets as Rand1%, Rand4%, ..., Rand99% respectively. Each dataset contains 200,000 images in total, and we investigate the distribution of samples from the generative models trained on Rand1%, Rand4%, ..., Rand99% respectively [1].

Evaluating Generative Models using Their Label Distributions as Summary Statistics: We propose to focus on high-level properties – labels in particular – of the learned generative model distribution. If the generated distribution and the real data distribution are to match, one necessary condition is that the *combinations* of the two distributions should be identical. Moreover, by leveraging digit classifiers with state-of-the-art performance, it is straightforward to obtain the labels of every generated sample with high confidence. This analysis also allows us to visualize the label distribution of the samples and compare with the real distribution and immediately identify whether the generated distributions exhibit certain pathological properties such as mode collapse [11]. Similar approaches have been adopted in [12, 13] where labels are used to approximately demonstrate mixing of Markov chains.

Computing the combination distribution: We split the generated samples into three 28×28 images. The three images are then classified by a pretrained 10-way MNIST classifier C with $< 1\%$ error, resulting in one prediction for each image. Then we combine the three prediction to obtain 1 of the 1000 combinations as the combined-label for the generated image. This allows us to obtain a histogram for the 1000 combinations through an empirical estimate. Ideally, the generated combination distribution (denoted q) should match the true combination distribution p, which has probability $1/\ell$ over the ℓ selected combinations C_i, and zero probability over others.

Evaluating the combination distribution: We consider three methods to compare the distances between q and p. Cross entropy (CE) evaluates the average number of bits needed to identify an event in p using coding scheme optimized for q; Earth Mover's Distance (EMD) measures the minimum transport from q to p. These two methods provide a holistic view about the "distance" between p and q, yet they do not distinguish errors from assigning the wrong probabilities to the true labels and errors from assigning probability to false labels. Therefore, we propose another performance measure that draws insights from the Area Under the ROC Curve (AUC).

[1] Training details in Appendix B.

% of combinations	1	4	16	32	50	84	96	99
CE	0.217	0.363	0.683	1.10	0.98	0.39	0.33	0.30
EMD	0.147	0.211	0.336	0.283	0.201	0.130	0.128	0.121
AUC	1.000	1.000	0.987	0.601	0.543	0.551	0.565	0.611
$L(\cdot)$	10	40	160	320	500	193	73	43

Table 1: **Upper:** Performance measures for the generated distribution when trained on datasets with different portions of randomly preselected combinations. For CE and EMD, lower is better. For AUC, higher is better. **Lower**: The predicted code length $L(\cdot)$ proposed in Section 3, where $c = 10, d = 1$.

Consider a classification problem over the combinations C_i where the goal is to distinguish whether some combination is in the support of p. Therefore the ground truth label for C_i is $\mathbb{I}(p(C_i) > 0)$. We treat $q(C_i)$ as predictions from the classifier; if $q(C_i)$ is higher, then C_i is more likely to be predicted as positive according to q. The ROC curve plots the true positive rate against the false positive rate at various threshold settings; AUC simply measures the area under this curve which considers the rank of the $q(C_i)$ values. This has the advantage of being independent of the specific threshold / portion of selected combinations.

Table 1 contains the chosen measures of distances between q and p, over training sets with a growing number of digits combinations (modes) [2]. Increasing the number of combinations in the training set, as the number of combinations grows from 1% to 32%, CE and EMD increase while AUC decreases (with the exception of EMD at 32%). This difference in p and q indicates that the generator is either 1) not producing some combinations that were present in the training set (mode collapse) or 2) producing samples with digit combinations that were not present in the training set C_i it was trained on. However, the opposite occurs when the portion of combinations is large: increasing the portion of combinations from 50% to 99% actually decreases CE and EMD, while increasing AUC. In this case, *it appears that the model actually performs better with more combinations (modes) in the true distribution!*

This behavior is unexpected in two ways. On the one hand, it has been shown that neural networks are able to overfit to (or memorize) random labels in supervised learning, yet in the case of GANs, the generated distribution seems to generalize to certain combinations that it has not discovered. On the other hand, it seems that GANs does not always perform worse when there are more combinations/modes in the true distribution, which is contrary to the common belief that GANs would perform worse when there are more modes. While this cannot be simply explained by the "missing modes phenomenon", some of these empirical results can be explained by our following theory based on Minimal Description Length. Intuitively, if the distribution is harder to encode, it is harder for GANs to learn that distribution. As a consequence, GAN will learn an imperfect distribution, covering either more or less combinations than what was present in the training set.

3 Measuring the Complexity of Label Distributions

The Minimum Description Length (MDL, [9]) framework provides a principled way to measure the complexity of the data/distribution. Under a set of allowed codes, there exists a computable minimum encoding length for the data. The minimum encoding length can then be utilized to determine the complexity of the distribution in question. Natural distributions are generally hard to encode and interpret in terms of Minimal Description Length because it is unclear as to what computer language / symbols should be used in order to represent the distributions. However, this can be made easier for the case of label distributions in our case. To further simplify things, we assume that all the labels within the support of the distribution are equally distributed. That is, for some label distribution p: $p(\ell) = 1/|\text{supp}(p)|, \forall \ell \in \text{supp}(p)$.

Thus we reduce the problem of describing the distribution p to describing its support set $\text{supp}(p)$. We denote $\mathcal{S}$ as the support set with elements $(X_1, \ldots, X_N)$, where $X_i \in \mathcal{X}_i, \forall i \in \{1, \ldots, N\}$. Let

[2] We show more detailed figures in the Appendix.

$L(\cdot)$ denote the minimum code length for any set. We then assume one particular encoding scheme that is intuitive and satisfies the following conditions:

- $\forall X_i, L(\{X_i\}) = 1$ (*one*)
- $\forall i, L(\mathcal{X}_i) = c$ for some constant $1 < c < |\mathcal{X}_i|$; (*all*)
- $\forall \mathcal{S}_1, \mathcal{S}_2 \in \mathcal{S}$, $L(S_1 \cup S_2) \leq L(S_1) + L(S_2)$ (*and*)
- $\forall \mathcal{S}_1 \in \mathcal{S}$, $L(\mathcal{S} \setminus \mathcal{S}_1) \leq L(\mathcal{S}) + dL(\mathcal{S}_1) + 1$ for some constant $d \geq 1$ (*except*)
- $\forall \mathcal{S}_i \in \mathcal{X}_i, \mathcal{S}_j \in \mathcal{X}_j$, $L(\mathcal{S}_i \times \mathcal{S}_j) \leq L(\mathcal{S}_i) + L(\mathcal{S}_j) + 1$ (*product*).

Given a fixed model/encoding scheme, the code length of some set/distribution only depends on the set itself. According to this encoding scheme, we are able to intuitively identify several sets/distributions that are "easy" or "hard" to represent:

- A distribution with a few possible labels is easy to represent because it is possible to enumerate all the possible labels through *and* with a short code.
- A distribution with all but few labels should also be easy to represent, since it is possible for the program to represent with the *all* set and enumerate for all the exceptions through *except*.
- A distribution with a large number of random labels is hard to encode using the encoding scheme described.

So far, we have defined a model that allows us to evaluate the code length required to represent a distribution. In the case considered in this paper, $N = 3$ (digits per image) and $|\mathcal{X}_i| = 10$. And thus we are able to describe the code length for the random label distributions. Assuming $\mathcal{S}$ is some set of $g\%$ random labels

- From the *and* operation, we have $L(\mathcal{S}) \leq 10g$;
- From the *all* and *product* operations, we have $L(\mathcal{X}_1 \times \mathcal{X}_2 \times \mathcal{X}_3) \leq 2 + 3c$;
- From the *except* operation, we have $L(\mathcal{S}) \leq 3 + 3c + 100d(10 - g)$;
- Therefore, $L(\mathcal{S}) \approx \min(10g, 3 + 3c + 10d(100 - g))$ and $L(\mathcal{S})$ takes its maximum when
$$g^\star = (3 + 3c + 1000d)/(10 + 10d)$$

In fact, the minimal code length $L(\cdot)$ of the distribution is closely related to the empirical observations (see Table 1), which we relate to GAN's capability of learning it. When $g \leq g^\star$, the *and* operation would be used to encode the distribution; the distribution gets harder to encode with more labels, leading to worse performance in most cases (with the exception of 32 in EMD and 50 in CE). When $g \geq g^\star$, the *all* but *except* case is more dominant, the distribution is easier to encode, so the performance actually gets higher with more labels present. Hence, this MDL perspective would explain the surprising empirical phenomenons that we observe in Section 3. It also tells us how the generative model will generalize when it cannot encode the distributions with many labels – it will first attempt to encode the *all* case and then the individual *exceptions*. In fact, Figure 3 (in the Appendix) shows that the label distributions learned with $50\% - 99\%$ labels are very similar even when trained with a different set of labels.

4 Discussion

In this paper, we discuss the performance of GANs from another quantitative perspective – the distribution of labels. Label distributions are much easier to reason than natural ones; with sufficient label configurations, we can analyze the distribution of generated samples and compare them with true distributions, which allows us to identify particular problems in the process. In particular, we propose a theory that the ability of GANs to learn a certain distribution is closely related to the encoding length of that distribution, which is not necessarily related to the number of modes.

While the empirical evidence appears to support this theory, this investigation is far from complete. We make a rather strong assumption on the uniform distribution of labels and do not consider the distributions within the labels. Our analysis also fails to consider cases in which the labels are continuous (and potentially have an infinite number of configurations). Therefore, it is interesting to see a more formal description for the theory presented as well as more empirical evidence that validate this claim.

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[2] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[3] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," *arXiv preprint arXiv:1703.10593*, 2017.

[4] A. Odena, "Semi-supervised learning with generative adversarial networks," *arXiv preprint arXiv:1606.01583*, 2016.

[5] V. Kuleshov and S. Ermon, "Deep hybrid models: Bridging discriminative and generative approaches," *UAI*, 2017.

[6] Y. Li, J. Song, and S. Ermon, "Inferring the latent structure of human decision-making from raw visual inputs," *arXiv preprint arXiv:1703.08840*, 2017.

[7] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems*, pp. 4565–4573, 2016.

[8] V. N. Vapnik and V. Vapnik, *Statistical learning theory*, vol. 1. Wiley New York, 1998.

[9] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.

[10] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," *arXiv preprint arXiv:1611.03530*, 2016.

[11] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.

[12] J. Song, S. Zhao, and S. Ermon, "Generative adversarial training for markov chains," *ICLR 2017 (Workshop Track)*, 2017.

[13] J. Song, S. Zhao, and S. Ermon, "A-nice-mc: Adversarial training for mcmc," *arXiv preprint arXiv:1706.07561*, 2017.

[14] S. Arora and Y. Zhang, "Do gans actually learn the distribution? an empirical study," *arXiv preprint arXiv:1706.08224*, 2017.

[15] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," *arXiv preprint arXiv:1704.00028*, 2017.