Theoretical Insights into Memorization in GANs

Vaishnavh Nagarajan
Computer Science Department
Carnegie-Mellon University
Pittsburgh, PA 15213
`vaishnavh@cs.cmu.edu`

Colin Raffel
Google Brain
`craffel@google.com`

Ian J. Goodfellow
Google Brain
`goodfellow@google.com`

Abstract

While significant research has gone into developing Generative Adversarial Network (GAN) architectures and training procedures that produce high-quality images, very little is known about how and when GANs "generalize" from the training data (i.e., produce unseen images) as opposed to simply memorizing the training dataset. To answer this question, in this paper we take a new theoretical perspective: we cast memorization in GANs a *learning* task on the generator that corresponds to learning a mapping from latent space to the data space. Specifically, in this task the generator is trained to output replicas of the training data on a finite set of latent vectors sampled during GAN training. Then we theoretically analyze how likely it is that the generator outputs the training data even on "unseen" latent vectors after training. By taking this perspective, we shed insights into the dynamics of how GAN training can lead to memorization, and uncover subtle, implicit factors in GAN training that *discourage* memorization.

1 Introduction

GANs [4] have become a popular class of deep-network based generative models owing to their unprecedented success in producing high quality images [6, 9]. Most of the empirical and theoretical works in this area have focused on building architectures [12] and training procedures that lead to improved sample quality and/or exhibit more stability during training [13, 1, 5, 11, 7].

In this work we theoretically address a crucial but largely unstudied question in GANs: in what scenarios do GANs generalize and produce 'unseen' samples (samples not from the training dataset) vs. simply memorize the training dataset? We shed insights into this question by looking at GAN training from a previously unstudied angle. Specifically, we cast the phenomenon of *GAN-memorization* as a *learning* task $\mathscr{L}_{\text{mem}}$ on the generator[1], in which the generator learns a mapping from the latent space to the data space. We consider the scenario that the generator has been trained to produce only data from the the training dataset *although only on a finite set of latent vectors* randomly sampled during training. We then study how well the generator would have learned to memorize the training set i.e., output images only from the training dataset *even on unseen points in the latent vector space*. By taking this approach where GAN-memorization requires the generator to generalize well in the learning task $\mathscr{L}_{\text{mem}}$, we argue the following:

1. A generator can be guaranteed to memorize the training dataset (or more precisely, uniform convergence can be guaranteed in $\mathscr{L}_{\text{mem}}$) only if it is trained on "sufficiently many" random latent vectors during training.
2. The longer that we train a generator (i.e., the more random latent vectors it is trained on), the stronger the level of GAN-memorization that it achieves.

Finally, through toy experiments, we demonstrate the utility our perspective of casting GAN-memorization as a learning task on the generator. In these experiments, we consider a natural modification of GAN training in which we pick

[1]We coin "GAN-memorization" to distinguish it from standard notions of generalization/memorization in supervised learning in the context of $\mathscr{L}_{\text{mem}}$, wherever possible.

Integration of Deep Learning Theories Workshop, 32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

a finite set of latent vectors before we begin training the GAN; we then train the GAN for a sufficiently long time but by sampling only from the pre-fixed finite set of latent vectors. By doing so, we observe a clear increase in GAN-memorization, when the number of fixed latent vectors is larger.

Related Work. There has been little theoretical work [3, 14, 2] studying generalization/memorization-related questions in GANs. Arora et al. [3] provide a formulation for generalization guarantees for GANs, and argue about the choice of the metric between distributions that generalization guarantees can be provided for. Their guarantees are based on an ϵ-net based argument; Zhang et al. [14] discuss generalization guarantees for a similar metric using Rademacher complexity-based arguments (which is the tool we use too). Arora and Zhang [2] provide a theoretically-motivated evaluation method (based on the birthday paradox) for measuring GAN-memorization.

Crucially, these works largely deal with understanding the effect of the discriminator and the size of the training data on generalization. Our work is orthogonal in that we focus on the generator in isolation, and study generalization in terms of the number of latent vectors that have been sampled during training – the role of which has never been considered so far. Our approach thus reveals nuanced factors that play a role in GAN-memorization, besides standard factors like the size of the discriminator architecture that have been explored in these prior works.

1.1 Preliminaries

Let $\mathbb{X}$ be the (image) data space, and $\mathbb{Z} = \mathbb{R}^K$ be the latent space. Let $\mathcal{X}$ be an unknown distribution over $\mathbb{X}$ and $\mathcal{Z}$ a known distribution (typically zero-centered Gaussian or uniform) in $\mathbb{Z}$. Recall that a GAN consists of a discriminator $D : \mathbb{X} \to \mathbb{R}$ and a generator $G : \mathbb{Z} \to \mathbb{X}$. Let ϕ be a monotone function, then the GAN objective and its many standard variants can be written as the following min-max game:

$$\min_G \max_D \mathbb{E}_{x\sim\mathcal{X}}[\phi(D(x))] + \mathbb{E}_{z\sim\mathcal{Z}}[\phi(1 - D(G(z)))] \tag{1}$$

In practice we only have access to a finite set of real data points S_{data}. Furthermore, we approximate the expectations above by replacing them with the empirical average on S_{data} and an empirical average over a finite set of latent vectors sampled from $\mathcal{Z}$, respectively. Typically, in each iteration, we draw an independent, finite set of latent vectors from $\mathcal{Z}$. We then train the generator and discriminator in tandem to find the equilibrium of the objective.

Objective (1) encourages GAN-memorization. It can be argued as it was done in the original paper [4] that, given sufficiently complex discriminator and generator models, the generator is encouraged to converge to an equilibrium that corresponds to the empirical distribution over S_{data}. The intuition is that a complex discriminator model would be able to tell apart all data in S_{data} from the generated data, as long as the generated data does not (closely) match S_{data}; thus, until the generator produces S_{data} exactly (or almost exactly), the system will be not be in equilibrium. In the following section we argue that, despite the fact that the objective is designed for convergence to the empirical distribution, due to the fact that the generator is trained only on a finite set of latent vectors, GAN-memorization is not an easy goal.

2 GAN-Memorization requires training on many latent vectors

In this section, we present our main result, which is that, under some conditions, GAN-memorization would be achieved only if we train the generator on sufficiently many latent vectors (to output S_{data} on those vectors). Our key idea is to formalize GAN-memorization as a generator learning task. First let S_{latent} be a finite set of latent vectors sampled from $\mathcal{Z}$. To simplify our discussion, we assume there exists an embedding $h : \mathbb{X} \to \mathbb{Z}'$ where $\mathbb{Z}' = \mathbb{R}^{K'}$ such that $h(x)$ for $x \sim \mathcal{X}$ is uniformly distributed in $[-1, 1]$ along each of the K' dimensions. We assume $K' = \Omega(\log m_{\text{data}})$ which is a reasonably small quantity. [2] We then choose a small constant[3] $\alpha > 0$ and say that G produces a datapoint identical to some x, at z, if $\|h(G(z)) - h(x)\| \leq \alpha$. Intuitively h can be thought of as a hash function for each datapoint and $\|h(G(z)) - h(x)\| \leq \alpha$ represents a "collision" between the hashes for two datapoints.

Note that the choice of the uniform distribution in $\mathbb{Z}'$ is flexible and the following results in Theorems 2.1 and 2.2 would hold good even when $\mathbb{Z}'$ is say a Gaussian distribution, or a uniform distribution over the Hamming space $\{0, 1\}^{K'}$. In the latter case, one could set $\alpha = 1$, so that $\|h(G(z)) - h(x)\| < \alpha$ can be thought of as an exact collision in this space.

[2] Then, with high probability over the draws of $\log m_{\text{data}}$, the points in S_{data} are sufficiently far apart by a distance that is close to the expected distance of $2\sqrt{K'}$, see Lemma A.1

[3] The reader can think of α to be much smaller than the expected inter-point distance of $\mathcal{O}(\sqrt{K'})$

The generator learning task $\mathscr{L}_{\text{mem}}$. We consider a learning algorithm that takes as input a set S_{data} of m_{data} training data drawn i.i.d from $\mathcal{X}$ and a set S_{latent} of m_{latent} latent vectors drawn i.i.d from $\mathcal{Z}$ (where $m_{\text{latent}} \geq m_{\text{data}}$). The learner then searches through a sufficiently large set of generators $\mathcal{G}_{\text{mem}}$ and picks a $G^\star \in \mathcal{G}_{\text{mem}}$ for which the distribution of $G^\star(z)$ induced by the empirical distribution over S_{latent} is identical (as defined above) to the empirical distribution over S_{data}. In other words, the learner has managed to produce a generator that has memorized the training data on the finite set of latent vectors S_{latent}. To state this formally, let us define the function $M_\alpha(x, S_{\text{data}}) = \mathbf{1}[\min_{x' \in S_{\text{data}}} \|h(x) - h(x')\|_2 \leq \alpha]$ which is one only when $x \in \mathbb{X}$ closely matches a point in the training dataset. Then, we have that for any S_{data} and S_{latent}, $\mathcal{G}_{\text{mem}}$ is sufficiently large that there exists $G^\star \in \mathcal{G}_{\text{mem}}$ that satisfies $M_\alpha(G^\star(z), S_{\text{data}}) = 1$ for all $z \in S_{\text{latent}}$.

To guarantee that the generator $G^\star$ learned on S_{data} and S_{latent} would have *learned to memorize* the training data on unseen latent vectors, one would typically show *uniform convergence*. That is, one would show that on most draws of S_{data} and S_{latent}, the difference between the averages of $M_\alpha(G(z), S_{\text{data}})$ over $z \in S_{\text{latent}}$ and $z \sim \mathcal{Z}$ is small *for all $G \in \mathcal{G}_{\text{mem}}$* (and not just the $G^\star$ corresponding to that draw of S_{data} and S_{latent}). Below, we show that one cannot provide such a uniform convergence guarantee without suffering from a generalization error of $\Omega(m_{\text{data}}/m_{\text{latent}})$. At a high level, this is because in order to be able to memorize m_{data} many datapoints, the learning algorithm has to ensure that $\mathcal{G}_{\text{mem}}$ has a sufficiently large representational capacity, thus making the uniform convergence guarantee harder. Our result assumes that a regularity condition on the functions in $\mathcal{G}_{\text{mem}}$ described in Appendix A.

Theorem 2.1. *Under Regularity Condition I, we have that for any S_{data}, with probability at least $1 - \delta$ over the draws of S_{latent}, uniform convergence does not hold in $\mathscr{L}_{\text{mem}}$:*

$$\sup_{G \in \mathcal{G}_{\text{mem}}} \left| Pr_{z \sim \mathcal{Z}}[M_\alpha(G(z), S_{\text{data}}) = 1] - \sum_{z \in S_{\text{latent}}} \frac{M_\alpha(G(z), S_{\text{data}})}{m_{\text{latent}}} \right| = \Omega\left(\frac{m_{\text{data}}}{m_{\text{latent}}}\right) - \mathcal{O}\left(\sqrt{\frac{\log \frac{1}{\delta}}{m_{\text{latent}}}}\right)$$

The above result suggests that to guarantee GAN-memorization (through uniform convergence) on all but at most some small constant mass of the latent space distribution, one needs to train the generator to reproduce the training dataset on as many as $\Omega(m_{\text{data}})$ latent vectors. We argue that this implies that making the generator "learn to memorize" the training data is a harder task, than say, making it "learn to output realistic, but unseen data". Specifically, in Appendix A.4 we show that it is reasonable to expect that by training the generator to output realistic data[4] (that are not copies of the training data) on as few as $\mathcal{O}(1)$ many latent vectors, the generator can be guaranteed to produce realistic data on all but at most a small constant mass of the latent space.

We note that in practice, GAN-memorization may be even harder (i.e., one might need many more samples) since the dependence on m_{data} in Theorem 2.1 hides other multiplicative factors that depend on the prior induced by the architecture. For example, if the model was such that it learns to map a non-zero mass of the K-dimensional-latent space to a particular training datapoint only if it is trained to map at least K latent vectors (that do not lie on a $K-1$-dimensional subspace) to that datapoint, we would need $m_{\text{latent}} = \Omega(K m_{\text{data}})$ many latent vectors during training to achieve GAN-memorization. On the other hand, if the model is biased to learn only piece-wise constant functions, it may need only m_{data} latent vectors (but this does not seem to be the case for neural networks in practice as seen from the experiments in the following section).

Next, we conversely state the following generalization guarantee for $\mathscr{L}_{\text{mem}}$ that lower bounds the proportion of the latent space on which the generator would have memorized the training data:

Theorem 2.2. *Under Regularity Condition II, with probability at least $1 - \delta$ over the draws of S_{latent}:*

$$Pr_{z \sim \mathcal{Z}}[M_{2\alpha}(G^\star(z), S_{\text{data}}) = 1] > 1 - \tilde{\mathcal{O}}\left(\frac{m_{\text{data}}}{\alpha^2} \sum_{k=1}^{K'} \mathcal{R}_{m_{\text{latent}}}\left(h^{(k)} \circ \mathcal{G}_{\text{mem}}\right)\right)$$

Here, $\mathcal{R}(\cdot)$ corresponds to the Rademacher complexity (see Appendix) which usually scales as $1/\sqrt{m_{\text{latent}}}$. The key highlight of this result is that the guarantee scales inversely with α i.e., smaller the α, the larger is the generalization error in $\mathscr{L}_{\text{mem}}$. Effectively this means that by training the generator on more latent vectors (i.e., run GAN training for longer), we can achieve stronger levels of GAN-memorization.

3 Experiments

To demonstrate the validity of casting GAN-memorization as a learning problem, we consider the toy dataset of 7 regularly arranged datapoints from [8]. Here, we propose studying a natural modification of the GAN algorithm where

[4]A generator would output unseen, realistic data when the discriminator is not overfit on the training data.

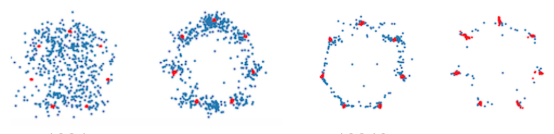


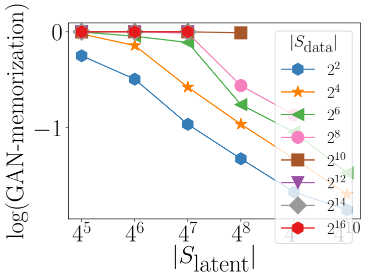
Figure 1: **Effect of number of latent vectors fixed for training on GAN-memorization:** The above figure shows the distribution learned by the GAN for different runs corresponding to varying magnitudes of m_{latent} (shown below each figure). The red points correspond to the generator's output on the training latent vectors S_{latent}; as evident, the generator has been trained to memorize the 7 real datapoints on S_{latent}. The blue points correspond to the output on $z \sim \mathcal{Z}$. As we increase m_{latent}, the distribution of blue points varies from a single blob covering all the datapoints, to a donut-like distribution, and finally to the precise empirical distribution.

we first draw a finite set of latent vectors S_{latent} before training; we then train the GAN by sampling latent vectors only from this fixed set, for a sufficiently long time until the output of the generator on the fixed latent vectors (marked in red in Figure 1) matches the real data. We observe that the output distribution of the GAN on unseen latent vectors (marked in blue) increases in the level of GAN-memorization for larger m_{latent}.

As pointed out in the discussion under Theorem 2.1, observe that we need as many as ≈ 50000 latent vectors in order to memorize 7 datapoints, suggesting that this neural network architecture has a prior that demands effectively ≈ 5000 latent vectors per real datapoint during training for GAN-memorization. This suggests that in practice, to memorize a dataset of say 50000 datapoints, we would have to train the generator for a sufficiently long time until it memorizes the dataset on as many as 10^8 randomly sampled latent vectors.

We also perform experiments in a setup where we train a generator in isolation to learn a pre-defined mapping from the latent space to datapoints S_{data} chosen almost randomly a high dimensional space (we present the exact technical details in the appendix). Here, we observe that the level of memorization indeed increases significantly with increase in $|S_{\text{data}}|$.While the generator in this setting is not exactly trained in the same setting as that of a GAN, our experiments are valuable in understanding the statistical obstacles in training a generator to memorize the datapoints.

We note that in the appendix, in Figure 4 we show some additional prelimiary experiments demonstrating a relationship between the batch size of the generator and GAN-memorization; we argue how this observation reinforces our approach of viewing GAN-memorization in itself as a learning task. We leave further investigation of this for future work.



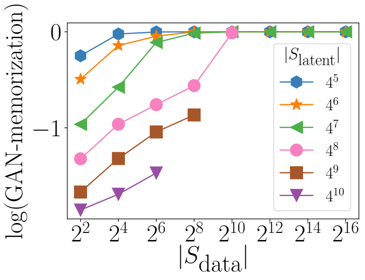


Figure 2

Figure 3: **Experiments in high-dimensional setup**: In the left, we plot how the value of $Pr_{z\sim\mathcal{Z}}[M_{2\alpha}(G^\star(z), S_{\text{data}}) = 1]$ varies with $|S_{\text{latent}}|$ and on the right how it varies with $|S_{\text{data}}|$.

4 Conclusion

In this work, we showed that casting GAN-memorization as a learning task provides valuable insights into understanding when and how GANs memorize the training datapoints. Our work seems to suggest that in practice, when the generator architecture is sufficiently large, one should avoid training the GAN for too many iterations (if in each iteration a fresh random latent vector is picked), or alternatively, one could train the GAN for a long time only as long as the latent vectors are picked from a finite set that is not too large. In the future, it would be important to understand empirically whether state-of-the-art GAN architectures do suffer from GAN-memorization for the usual number of epochs that they are trained on, and if so, whether fixing the latent vectors helps prevent GAN-memorization.

References

[1] 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*, pages 214–223, 2017.

[2] Sanjeev Arora and Yi Zhang. Do gans actually learn the distribution? an empirical study. In *Sixth International Conference on Learning Representations (ICLR)*, 2018.

[3] 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*, volume 70 of *Proceedings of Machine Learning Research*, pages 224–232, 2017.

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680. 2014.

[5] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein GANs. In *Thirty-first Annual Conference on Neural Information Processing Systems (NIPS)*. 2017.

[6] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. 2018.

[7] Lars M. Mescheder, Sebastian Nowozin, and Andreas Geiger. The numerics of gans. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, 2017.

[8] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. In *Fifth International Conference on Learning Representations (ICLR)*. 2017.

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

[10] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. Adaptive computation and machine learning. MIT Press, 2012. URL `http://mitpress.mit.edu/books/foundations-machine-learning-0`.

[11] Vaishnavh Nagarajan and J. Zico Kolter. Gradient descent GAN optimization is locally stable. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, 2017.

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

[13] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In *Advances in Neural Information Processing Systems 29*, pages 2234–2242. 2016.

[14] Pengchuan Zhang, Qiang Liu, Dengyong Zhou, Tao Xu, and Xiaodong He. On the discrimination-generalization tradeoff in gans. In *Sixth International Conference on Learning Representations (ICLR)*, 2018.

A Appendix

We first prove the following (standard) result which states that the points in S_{data} are sufficiently far apart in $\mathbb{Z}'$ when $K' = \Omega(\log m_{\text{data}})$ i.e., when the number of latent dimensions in the data is not too small. We will find this result useful in our subsequent proofs.

Lemma A.1. *With probability at least* $1 - 1/e^{\Omega(K')}$ *over the draws of* $S_{\text{data}} = \{x_1, x_2, \cdots, \}$, *for all* $i \neq j$,

$$\|h(x_i) - h(x_j)\| = \Omega(\sqrt{K'})$$

Proof. For each $i \neq j$, the squared distance $\|h(x_i) - h(x_j)\|^2 = \sum_{k=1}^{K'} (h^{(k)}(x_i) - h^{(k)}(x_j))^2$ is essentially the sum of K' independent random variables from the support $[0, 2]$ with mean $\Theta(1)$. Then, by applying the Hoeffding bound (see Lemma C.4), we will get that with probability $1 - \delta/m_{\text{data}}^2$ over the draws of x_i and x_j:

$$\|h(x_i) - h(x_j)\|^2 = \Omega(K') - \sqrt{K' \ln \frac{2m_{\text{data}}^2}{\delta}}$$

For $\ln m_{\text{data}} = \mathcal{O}(K')$ and $\ln 1/\delta = \mathcal{O}(K')$, the second quantity above would be $\mathcal{O}(K')$. In other words, $\|h(x_i) - h(x_j)\|^2 = \Omega(K')$. By applying this bound on all the $\mathcal{O}(m_{\text{data}}^2)$ pairs of points separately, followed by a union bound we get the result above.

□

A.1 Regularity Conditions

Below we state our regularity conditions on the generators in $\mathcal{G}_{\text{mem}}$.

Regularity Condition I. Let $S_{\text{data}} = \{x_1, x_2, \cdots\}$ and $S'_{\text{data}} = \{x'_1, x'_2, \cdots\}$ be two 'close' sets such that for all i, $h(x_i) - h(x'_i) \leq c\sqrt{K'}$ where c is some small constant. Let $G^\star$ and G' be the generators learned on these datasets, given the training latent vectors S_{latent}; then, we assume that for all $z \in S_{\text{latent}}$, if for some i, $\|h(G^\star(z)) - h(x_i)\| \leq \alpha$ then $\|h(G'(z)) - h(x'_i)\| \leq \alpha$.

That is, every $z \in S_{\text{latent}}$ that is closely mapped to x_i by $G^\star$, is also mapped to x'_i by G'. Or in other words, for a fixed S_{latent}, under small changes to the dataset S_{data}, the mapping from S_{latent} to S_{data} is preserved by the generator learned by the algorithm, upto the small changes in S_{data}.

Regularity Condition II. For all $G \in \mathcal{G}_{\text{mem}}$, for all $z \in \mathbb{Z}$, for all $k \leq K'$, $|h^{(k)}(G(z))| = \mathcal{O}(1)$ i.e., the output of every generator is bounded by a constant in each dimension of $\mathbb{Z}'$.

A.2 Proof Theorem 2.1

Let $S_{\text{latent}} = \{z_1, z_2, \cdots, z_{m_{\text{latent}}}\}$. The bound follows from a standard Rademacher complexity bound (Lemma C.3) which implies that there exists $G \in \mathcal{G}_{\text{mem}}$ for which:

$$Pr_{z \sim \mathcal{Z}}[M_\alpha(G(z), S_{\text{data}}) = 1] < \frac{\sum_{z \in S_{\text{latent}}} M_\alpha(G(z), S_{\text{data}})}{m_{\text{latent}}} + \mathcal{O}\left(\sqrt{\frac{\log \frac{1}{\delta}}{m_{\text{latent}}}}\right)$$
$$- \mathbb{E}_{S_{\text{latent}}} \mathbb{E}_{\boldsymbol{\sigma}} \left[\frac{1}{m_{\text{latent}}} \sup_{G \in \mathcal{G}_{\text{mem}}} \sum_{j=1}^{m_{\text{latent}}} \sigma_j M_\alpha(G(z_j), S_{\text{data}}) \right]$$

where, σ_j are Rademacher variables i.e., these are independent random variables with equal probability of being 1 and -1. We now show that the numerator of the expectation in the last term is $\Omega(m_{\text{data}})$, thereby completing the proof.

Let $G^\star \in \mathcal{G}_{\text{mem}}$ be the generator learned by the algorithm on S_{latent}. For each $\boldsymbol{\sigma}$, we will construct a different dataset $S^{\boldsymbol{\sigma}}_{\text{data}} = \{x'_1, x'_2, \cdots, \}$, such that the generator, say $G^{\boldsymbol{\sigma}} \in \mathcal{G}_{\text{mem}}$, learned on this dataset maximizes the summation within the expectation above. For each $x_i \in S_{\text{data}}$, let $S^{(i)}_{\text{latent}} \subset S_{\text{latent}}$ be the subset of latent vectors that $G^\star$ maps to x_i. Note that by definition of $G^\star$, for each i, an equal proportion of the latent vectors in S_{latent} would be mapped to each of the datapoints in S_{data}. Therefore, we have that $|S^{(i)}_{\text{latent}}| = \frac{m_{\text{latent}}}{m_{\text{data}}}$. Now, if the σ_j's corresponding to $S^{(i)}_{\text{latent}}$ have a majority of +1's, we let $x'_i = x_i$. Otherwise, we let x'_j to be any x'_j that satisfies $\|h(x'_j) - h(x_j)\| > 2\alpha$ and $\|h(x'_j) - h(x_j)\| \le c\sqrt{K'}$. [5]

recall that we assumed α to be a constant smaller than $\sqrt{K'}$, so $2\alpha < c\sqrt{K'}$, such an x'_j should exist).

Then observe that S_{data} and $S^{\boldsymbol{\sigma}}_{\text{data}}$ satisfy the requirements of Regularity Condition I. Then, by applying this condition, we have that for each i, for all $z \in S^{(i)}_{\text{latent}}$, $\|h(G^{\boldsymbol{\sigma}}(z)) - h(x'_i)\| \le \alpha$ i.e., $G^{\boldsymbol{\sigma}}$ maps $S^{(i)}_{\text{latent}}$ to x'_i. Given this, we next evaluate $M_\alpha(G^{\boldsymbol{\sigma}}(z), S_{\text{data}})$ for each $z \in S_{\text{latent}}$.

If the σ_j's corresponding to $S^{(i)}_{\text{latent}}$ have a majority of +1's, we have that since $x'_i = x_i$, by the regularity assumption, for all $z \in S^{(i)}_{\text{latent}}$, $\|h(G^{\boldsymbol{\sigma}}(z)) - h(x_i)\| = \|h(G^{\boldsymbol{\sigma}}(z)) - h(x'_i)\| \le \alpha$. Thus, $M_\alpha(G^{\boldsymbol{\sigma}}(z), S_{\text{data}}) = 1$.

On the other hand, when the majority is -1, we argue that for all $z \in S^{(i)}_{\text{latent}}$, $M_\alpha(G^{\boldsymbol{\sigma}}(z), S_{\text{data}}) = 0$ i.e., for all t, $\|h(G^{\boldsymbol{\sigma}}(z)) - h(x_t)\| > \alpha$. We first show this for $t = i$ and then for $t \ne i$. For $t = i$, we first note that we picked x'_i such that $\|h(x'_i) - h(x_i)\| > 2\alpha$. By the regularity assumption, we also have $\|h(G^{\boldsymbol{\sigma}}(z)) - h(x'_i)\| \le \alpha$. By the triangle inequality, we then have that $\|h(G^{\boldsymbol{\sigma}}(z)) - h(x_i)\| > \alpha$.

Now, consider $t \ne i$. From Lemma A.1, we have that, for all $t \ne i$, $\|h(x_i) - h(x_t))\| = \Omega(\sqrt{K'})$. Since we picked x'_i such that $\|h(x'_i) - h(x_i)\| \le c\sqrt{K'}$, for a sufficiently small constant c, we would have by triangle inequality that $\|h(x'_i) - h(x_t)\| = \Omega(\sqrt{K'})$. Then, from the regularity assumption and by the triangle inequality, we get that for all $k \ne i$, for all $z \in S^{(i)}_{\text{latent}}$, $\|h(G^{\boldsymbol{\sigma}}(z)) - h(x_t))\| = \Omega(\sqrt{K'}) > \alpha$.

Thus, from the above two cases, we have that when the majority of σ_j corresponding to $S^{(i)}_{\text{latent}}$ is -1, $M_\alpha(G^{\boldsymbol{\sigma}}(z), S_{\text{data}}) = 0$ for all $z \in S^{(i)}_{\text{latent}}$.

Finally, assuming that $m_{\text{latent}}/m_{\text{data}}$ is an odd integer, we can argue that there is always a strict majority of the positive/negative Rademacher variables in $S^{(i)}_{\text{latent}}$ for each i; therefore, for $G^{\boldsymbol{\sigma}}$ the corresponding summation over the σ_j's of $S^{(i)}_{\text{latent}}$ either equals zero (when the majority is -1) or is at least 1 (when the majority is 1). Thus, the overall summation in the numerator would be at least the number of i's such that $S^{(i)}_{\text{latent}}$ corresponds to a majority of positive Rademacher variables. For each such i, this would happen with probability $1/2$ over the draws of $\boldsymbol{\sigma}$ and by linearity of expectations, the sum in the numerator would evaluate to $m_{\text{data}}/2$.

A.3 Theorem 2.2

The given bound is inspired by a standard margin-based approach to deriving Rademacher complexity bounds where the generalization guarantee on a non-smooth 0-1 loss function is derived via a generalization guarantee on a 'smoother' surrogate. Our analysis will use a smoother formulation of M, denoted by M'. We begin by focusing on a particular point $x' \in S_{\text{data}}$, and define $M'_\alpha(x, \{x'\})$ as follows:

$$M'_\alpha(x, \{x'\}) = \begin{cases} 1 & \|h(x) - h(x')\|^2 \le \alpha^2 \\ \frac{4}{3} - \frac{1}{3\alpha^2}\|h(x) - h(x')\|^2 & \|h(x) - h(x')\|^2 \in [\alpha^2, 4\alpha^2] \\ 0 & \text{otherwise} \end{cases}$$

We will aggregate our analysis across all $x' \in S_{\text{data}}$ towards the end of this proof.

We first upper bound the following Rademacher complexity:

$$\mathbb{E}_{S_{\text{latent}}}\mathbb{E}_{\boldsymbol{\sigma}}\left[\frac{1}{m_{\text{latent}}} \sup_{G \in \mathcal{G}_{\text{mem}}} \sum_{j=1}^{m_{\text{latent}}} \sigma_j M'_\alpha(G(z_j), \{x'\})\right]$$

[5]Note that it would be possible to find such an x'_j because α is small constant and therefore $2\alpha < c\sqrt{K'}$. Furthermore, the distribution $h(x)$ for $x \sim \mathcal{X}$, is a uniform distribution over $[-1,1]^{K'}$, hence, there must be points in $[-1,1]^{K'}$ that are within $c\sqrt{K'}$ distance of $h(x_j)$, and there must exist a non-zero mass of $\mathcal{Z}$ in $\mathbb{X}$ that maps to these points.

since $M'_\alpha(G(z_j), \{x'\})$ is $1/3\alpha^2$-Lipschitz in $\|h(G(z_j)) - h(x')\|^2$, by Talagrand's lemma (see Lemma C.2) we have the following upper bound:

$$\le \frac{1}{3\alpha^2}\mathbb{E}_{S_{\text{latent}}}\mathbb{E}_{\boldsymbol{\sigma}}\left[\frac{1}{m_{\text{latent}}}\sup_{G\in\mathcal{G}_{\text{mem}}}\sum_{j=1}^{m_{\text{latent}}}\sigma_j\|h(G(z_j)) - h(x')\|^2)\right]$$

the summation over K' within the sup, can be brought outside:

$$\le \frac{1}{3\alpha^2}\sum_{k=1}^{K'}\mathbb{E}_{S_{\text{latent}}}\mathbb{E}_{\boldsymbol{\sigma}}\left[\frac{1}{m_{\text{latent}}}\sup_{G\in\mathcal{G}_{\text{mem}}}\sum_{j=1}^{m_{\text{latent}}}\sigma_j(h^{(k)}(G(z_j)) - h^{(k)}(x'))^2\right]$$

From Regularity Condition II, the embedding $|h^{(k)}(G(z_j))|$ is bounded by a constant $\mathcal{O}(1)$, and since $|h^{(k)}(x')| \le 1$ we have that $(h^{(k)}(G(z_j)) - h^{(k)}(x'))^2$ is $\mathcal{O}(1)$-Lipschitz in $h^{(k)}(G(z_j))$. Then, by the Talagrand's Contraction Lemma C.2, we get:

$$\le \mathcal{O}\left(\frac{1}{\alpha^2}\sum_{k=1}^{K'}\mathbb{E}_{S_{\text{latent}}}\mathbb{E}_{\boldsymbol{\sigma}}\left[\frac{1}{m_{\text{latent}}}\sup_{G\in\mathcal{G}_{\text{mem}}}\sum_{j=1}^{m_{\text{latent}}}\sigma_j h^{(k)}(G(z_j))\right]\right)$$
$$\le \mathcal{O}\left(\frac{1}{\alpha^2}\sum_{k=1}^{K'}\mathcal{R}_{m_{\text{latent}}}\left(h^{(k)}\circ\mathcal{G}_{\text{mem}}\right)\right)$$

which corresponds to the Rademacher complexity term present in the main theorem. Then, we have by a standard result (see Lemma C.1) that the following bound involving $x' \in S_{\text{data}}$ holds with probability at least $1 - \delta/m_{\text{data}}$ over the draws of S_{latent}:

$$\mathbb{E}_{z\sim\mathcal{Z}}[M'_\alpha(x, \{x'\})] \ge \frac{1}{m_{\text{latent}}}\sum_{z\in S_{\text{latent}}}M'_\alpha(x, \{x'\}) - \tilde{\mathcal{O}}\left(\frac{1}{\alpha^2}\sum_{k=1}^{K'}\mathcal{R}_{m_{\text{latent}}}\left(h^{(k)}\circ\mathcal{G}_{\text{mem}}\right)\right)$$

Now, to arrive at the final bound in terms of the non-smooth function M, we first note that whenever $M_{2\alpha}(x, \{x'\}) = 0$, it means that $M'_\alpha(x, \{x'\}) = 0$. Furthermore, when $M_{2\alpha}(x, \{x'\}) = 1$ we have $M_{2\alpha}(x, \{x'\}) \ge M'_\alpha(x, \{x'\})$ because $M'_\alpha(x, \{x'\})$ is always bounded above by one. Therefore, we can upper bound the left hand side above with $Pr[M_{2\alpha}(x, \{x'\}) = 1]$. Similarly, we know that when $M_\alpha(x, \{x'\}) = 1$ it means that $M'_\alpha(x, \{x'\}) = 1$. Thus, we can replace the first term on the right hand side with the proportion of training latent vectors for which $M_\alpha(G(z), \{x'\}) = 1$, which we know is $m_{\text{latent}}/m_{\text{data}}$. Hence, we get:

$$Pr_{z\sim\mathcal{Z}}[M_{2\alpha}(x, \{x'\}) = 1] \ge \frac{1}{m_{\text{data}}} - \mathcal{O}\left(\frac{1}{\alpha^2}\sum_{k=1}^{K'}\mathcal{R}_{m_{\text{latent}}}\left(h^{(k)}\circ\mathcal{G}_{\text{mem}}\right)\right) \tag{2}$$

Finally, we apply the above bound individually for each $x' \in S_{\text{data}}$ so that each holds with probability at least $1 - \frac{\delta}{m_{\text{data}}}$. Then, since the datapoints in S_{data} are at least $\Omega(\sqrt{K'}) > \alpha$ distance apart from Lemma A.1, we will have from the triangle inequality that for any x, $M_{2\alpha}(x, \{x'\})$ can be 1 for at most one point $x' \in S_{\text{data}}$. Thus,

$$Pr_{z\sim\mathcal{Z}}[M_{2\alpha}(x, S_{\text{data}}) = 1] = \sum_{x'\in S_{\text{data}}}Pr_{z\sim\mathcal{Z}}[M_{2\alpha}(x, \{x'\}) = 1] \tag{3}$$

From Equations 2 and 3, we get the result claimed.

A.4 Learning to produce realistic data requires training on fewer latent vectors

In this section, we consider a different learning task for the generator which we call $\mathscr{L}_{\text{realistic}}$. Informally, we consider the situation where the generator has been trained to output "realistic" data on finitely many latent vectors, and then study how realistic the output of the generator is, on unseen latent vectors in the latent space (we will formalize the notion of realisticity shortly). Our main argument here is that, when compared to the number of latent vectors required during training for achieving GAN-memorization, it is reasonable to expect that with fewer latent vectors during training, the generator learns to produce realistic examples on unseen latent vectors. In other words, learning to produce only replicas of S_{data} is harder than learning to produce unseen examples.

First, we formulate how realistic a sample x is, in terms of how far away it is from the origin in the embedding defined by $h(\cdot)$. Let $\beta \geq 1$ be a constant. Then, we define $R_\beta(x) = \mathbf{1}[h(x) < \beta\sqrt{K'}]$, which is one when the data point is realistic i.e., sufficiently close to the origin. Observe that the definition of R – unlike the definition of M – does not rely on S_{data}, as the notion of realisticity must be independent of the training data.

Now to formulate the learning task $\mathscr{L}_{\text{realistic}}$, we let $\mathcal{G}_{\text{small}}$ be a class of generator functions such that when the generator is trained to output realistic data on S_{latent}, it picks a $G^\star \in \mathcal{G}_{\text{small}}$. Note that to achieve this, the generator need not necessarily have to produce exact copies of its training data; thus it is reasonable to expect $\mathcal{G}_{\text{small}}$ to be much smaller than $\mathcal{G}_{\text{mem}}$. More concretely, we can hope that the Rademacher complexity of $\mathcal{G}_{\text{small}}$ does not scale with m_{data}, unlike $\mathcal{G}_{\text{mem}}$. In practice, $\mathcal{G}_{\text{small}}$ would correspond to the space of generator parameters that are reachable by the deep network in the initial phase of GAN training when the discriminator would have not yet overfit to the training data.

Now, we state the following generalization guarantee for $\mathscr{L}_{\text{realistic}}$ that lower bounds the proportion of the latent space distribution on which the generator produces realistic data:

Theorem A.2. *With probability at least* $1 - \delta$ *over the draws of* S_{latent}*, we have that:*

$$Pr_{z\sim\mathcal{Z}}[R_{2\beta}(G^\star(z)) = 1] = 1 - \tilde{\mathcal{O}}\left(\frac{1}{K'\beta^2}\sum_{k=1}^{K'}\mathcal{R}_{m_{\text{latent}}}\left(h^{(k)} \circ \mathcal{G}_{\text{small}}\right)\right)$$

Observe that the above generalization guarantee has no explicit dependence on m_{data} unlike in Theorems 2.1 and 2.2. Furthermore, it is reasonable to expect that the Rademacher complexity term above does not scale with m_{data} because $\mathcal{G}_{\text{small}}$ is a smaller class of generators. Thus, what we have is a generalization guarantee that does not scale with m_{data} and only scales as $1/\sqrt{m_{\text{latent}}}$. In other words, as long as we train the generator on $\mathcal{O}(1/\epsilon^2)$ many latent vectors, we are guaranteed that the generator would output realistic data on all but at most ϵ mass of the latent space.

Proof. The proof for this theorem parallels that of Theorem 2.2. We consider a smoother function $R'_\beta(x)$ which has the following form:

$$R'_\beta(x) = \begin{cases} 1 & \|h(x)\|^2 \leq K'\beta^2 \\ \frac{4}{3} - \frac{1}{3K'\beta^2}\|h(x)\|^2 & \|h(x)\|^2 \in [K'\beta^2, 4K'\beta^2] \\ 0 & \text{otherwise} \end{cases}$$

With a nearly identical analysis as in Theorem 2.2 using the above function, we arrive at the claimed bound. The only notable difference in the proof is the lack of dependence of the analysis on S_{data}. □

B Experiments

Effect of batch size on GAN-memorization. Interestingly, as a further evidence for the validity of our perspective of casting GAN-memorization as a learning task, we show in Figure 4 that, for fixed m_{latent}, *increasing the batch size of the generator results in decrease of GAN-memorization.* We argue that this is because smaller batch sizes are associated with better generalization of deep networks; naturally, in this case, it leads to better generalization in the learning task $\mathscr{L}_{\text{mem}}$, and hence stronger GAN-memorization. More generally, we hypothesize that any hyperparameter in the GAN training, that would result in better generalization for the learning task $\mathscr{L}_{\text{mem}}$ on the generator would result in increased GAN-memorization.

High-dimensional experiment details. For the plots in Figure 2, we choose $K = 80$. Next, for any given value of $|S_{\text{data}}|$, we pick a data space $\mathbb{X}$ of $\log|S_{\text{data}}|$ dimensions. From each of the $2^{|S_{\text{data}}|}$ hyper-quadrants in the space, we pick a random point from a uniform distribution within a hypercube in that hyper-quadrant and add it to S_{data}. Next, we define an explicit mapping from the latent space to S_{data}. We perform the mapping by considering the quadrant of the first $\log|S_{\text{data}}|$ dimensions of the latent vector and mapping it to the datapoint from the corresponding quadrant in the dataspace. We assume $h(\cdot)$ to be the identity mapping and train the generator to minimize the ℓ_2 error as per the mapping.

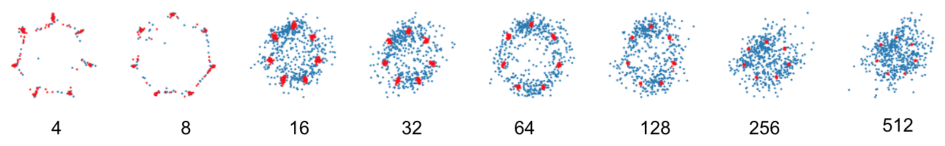


Figure 4: **Effect of generator batch-size on GAN-memorization:** The above figure shows the distribution learned by the GAN for different runs corresponding to varying batch sizes (specified under each image) given a fixed size of m_{latent}. Note that the level of GAN-memorization in these plots typically show a greater level of variance across different runs than in the experiments of Figure 1; we show the qualitatively most likely plot for each value of the batch size here.

C Useful Lemmas

In this section, we present some standard terminologies and results from Rademacher complexity theory. First, we define Rademacher complexity:

Definition C.1. *For any integer $m > 1$, the Rademacher complexity of a hypothesis class $\mathcal{F} : \mathbb{Z} \to [0,1]$, under a distribution $\mathcal{Z}$ over its inputs is denoted by:*

$$\mathcal{R}_m(\mathcal{F}) = \mathbb{E}_{S\sim\mathcal{Z}}\left[\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{f\in\mathcal{F}}\frac{1}{m}\sum_{j=1}^{m}\sigma_j f(z_j)\right]\right]$$

We then have the following uniform convergence based upper bound on the generalization error:

Lemma C.1. *Let $\mathcal{F} : \mathbb{Z} \to [0,1]$. For all $\delta > 0$, with probability $1-\delta$ over the draws of $S \sim \mathcal{Z}^m$, we have that:*

$$\sup_{f\in\mathcal{F}}\left|\mathbb{E}_{z\sim\mathcal{Z}}[f(z)] - \frac{1}{m}\sum_{z\in S} f(z)\right| \le 2\mathcal{R}_m(\mathcal{F}) + \sqrt{\frac{\log 2/\delta}{2m}},$$

Below we present Talagrand's contraction lemma from Lemma 4.2 in [10] which helps bound the Rademacher complexity of compositions of function classes:

Lemma C.2. *Let $\Phi : \mathbb{R} \to \mathbb{R}$ be an ℓ-Lipschitz function. Then:*

$$\mathcal{R}_m(\Phi \circ \mathcal{F}) \le \ell \cdot \mathcal{R}_m(\mathcal{F})$$

Finally, we present the lower bound on the generalization error:

Lemma C.3. *Let $\mathcal{F} : \mathbb{Z} \to [0,1]$. For all $\delta > 0$, with probability $1-\delta$ over the draws of $S \sim \mathcal{Z}^m$, we have that:*

$$\sup_{f\in\mathcal{F}}\left|\mathbb{E}_{z\sim\mathcal{Z}}[f(z)] - \frac{1}{m}\sum_{z\in S} f(z)\right| > \frac{1}{2}\mathcal{R}_m(\mathcal{F}) - \mathcal{O}\left(\frac{1}{\sqrt{m}}\right),$$

We finally state the Hoeffding bound:

Lemma C.4. *Let $z_1, z_2, \cdots, z_{K'}$ be drawn independently at random from a distribution with mean μ and support $[0,1]$. Then we have that:*

$$Pr\left[\left|\sum_{k=1}^{K'}(z_k - \mu)\right| \ge t\right] \le 2\exp\left(-\frac{2t^2}{K'}\right)$$