Implicit Bias towards the Kernel Regime Causes Mode Collapse in GANs

Anonymous Author(s)
Affiliation
Address
email

Abstract

Generative adversarial networks (GANs) are state-of-the-art generative models but are notoriously difficult to train and often suffer from mode collapse. Many GAN variants have been proposed to tackle it and have led to improvements, but the underlying mechanism driving the issue remains poorly understood. In this paper, we demonstrate that the implicit bias (IB) of the GAN generator hinders its ability to capture modes. First, we develop a theoretical characterization of optimal generator functions for 1-dimensional data distribution. We characterize the constraints that optimal polytonic generators have to satisfy, and show that they may possess more discontinuities between modes. Next, we hypothesize that three factors – overparameterization, larger weight initialization scale, and lower adaptivity in the optimizer – implicitly bias the generator towards the kernel regime, reducing its ability to approximate discontinuities and thus causing mode collapse and mixtures. We run experiments and perform causal analysis using instrumental variables to verify our hypothesis. Our analysis suggests IB in the kernel regime *causes* bad GAN performance. And finally, we empirically verify our findings on MNIST and show that we can improve GAN performance by using smaller networks or just reducing the generator's initialization scale.

1 Introduction

Generative adversarial networks (GANs). GANs are state-of-the-art generative models for a variety of tasks, such as generating images, audios, or videos, which can be used for data augmentation and creative works. However, they are notoriously difficult to train. A non-trivial amount of effort is required to design architecture, objectives, and training schemes to avoid issues like mode collapse and mode mixture [Gulrajani et al., 2017, Miyato et al., 2018, Wang et al., 2019] However, the underlying mechanism for mode collapse is still poorly understood. Is it due to the difficulty of its task? Or is it because G is parameterized by a neural network (NN)?

Generator as a transport map. The task of G is to approximate a transport map from a unimodal distribution, such as a uniform or Gaussian distribution, to a real data distribution, which in most cases is a multimodal distribution. The optimal transport map under some mild conditions is discontinuous [Villani, 2009], which is challenging for NNs to approximate [An et al., 2020a]. When G fails to approximate the discontinuities, mode mixture or mode collapse will arise. To tackle this, variants of GANs have been proposed, e.g. using a multimodal latent distribution [Xiao et al., 2018], using multiple generators [Hoang et al., 2018], and instead of approximating the transport map directly, approximating the Brenier potential of the transport map [An et al., 2020b].

The remaining mysteries. First, while people acknowledge that the optimal G is discontinuous, it is unclear what other property it has. Hence, we attempt to characterize them in 1D and find they can

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

be polytonic (See Def. 1), which may have more discontinuities than needed. We will describe the in detail in Section 3.

Second, while the proposed solutions above all reduce the difficulty of the task for G, each requires non-trivial modifications to the model architecture, objectives, etc. From a more fundamental point of view, we are curious why it is challenging for an overparameterized G, which has sufficiently expressive power to approximate discontinuities. What implicitly biases a NN away from doing so? This kind of mechanistic-level questions motivates a rich literature on the inductive bias (IB) of NN (see Section 2). The key is that overparameterization and large scale of initialization bias NNs to the kernel regime, where parameters barely change. However, in this line of work, less attention is given to GANs. Schaefer et al. [2020] discuss the impact of the implicit competitive regularization, showing that it introduces additional stable points for the training. However, ICR is primarily induced by competitive training, not by NNs themselves. And their analysis only focuses on the discriminator. Our hypothesis is that IB of G is in tension with the need of approximating discontinuities. The more it is in the kernel regime, the less capable it is to do so.

Main Contributions. In this work, we systematically study the impact of IB on GAN performance, by first characterizing the polytonicity in the optimal G, then confirming our hypothesis with experiments interpolating from the adaptive regime to the kernel regime, and finally identify the causal impact of IB. Our main contributions include:

1. We find that allowing polytonicity in a 1D generator (G) is seemingly more flexible but requires more discontinuities, especially in low data density regions. Experiments confirm that polytonicity does exist.
2. We perform causal analysis showing that G being more in the kernel regime causes worse GAN's performance, i.e. when G is in the kernel regime, achieved by overparameterization, large scale of initialization, or lack of adaptivity in optimizers.
3. Experiments on MNIST show similar results that overparameterization or large scale of initialization hurts GAN performance. By simply turning down the scale, we obtain better generated images. We argue that our findings provide insights on the implicit bias on GANs and suggest small width or scale for hyperparameter search.

Next, this work is presented as follows: in Section 2, we briefly review the literature on IB and discontinuities in G. In Section 3, we characterize the polytonicity in optimal G in 1D. In Section 4, we propose our hypothesis on how IB impacts GAN performance. And in Section 5, we present experimental results and causal analysis on the impact.

2 Related Work

Implicit bias (IB) in neural networks (NNs). It is shown in recent work that NNs exhibit hidden biases that are determined *implicitly*. The implicit bias (IB) of deep *linear* NNs (including Convnets) and deep nonlinear NNs in the kernel regime has been uncovered [Jacot et al., 2018, Gunasekar et al., 2018, Chizat et al., 2019, Sahs et al., 2020, Woodworth et al., 2020, Moroshko et al., 2020], but the IB of *nonlinear* architectures remains an area of active research [Novak et al., 2019, Sahs et al., 2020, Caro et al., 2020, Geiger et al., 2020, Baratin et al., 2021]. Nevertheless, it is clear that the IB of a NN usually takes the form of an implicit regularizer (IR) that depends on 1) the parameterization or architecture, 2) the learning algorithm or optimizer, and 3) the initialization scheme [Chizat et al., 2019, Woodworth et al., 2020]. A strong IR biases NNs towards the kernel or lazy-training regime, where the NN parameters remain around their initialization, allowing approximations such as linearizing the network using the neural tangent kernel (NTK) [Jacot et al., 2018, Lee et al., 2019]. The counterpart of this regime is called the adaptive or feature-training regime, where NNs are able to learn the features and align them quickly [Baratin et al., 2021], resulting in highly non-linear behavior. All these work focus on regression or classification tasks that only involve one NN. However, GANs involve 2 NNs playing a game: while the discriminator's (D) task seems to be still classification, the generator's (G) task is nothing similar to regression or classification. Hence, there is a need to systematically study the impact of IB in GANs.

Implicit bias in GANs. There is much less work on IB in GANs. Intuitively, there are at least 4 components whose IB needs our attention: 1) IB of G, 2) IB of D, 3) IB of competitive training on

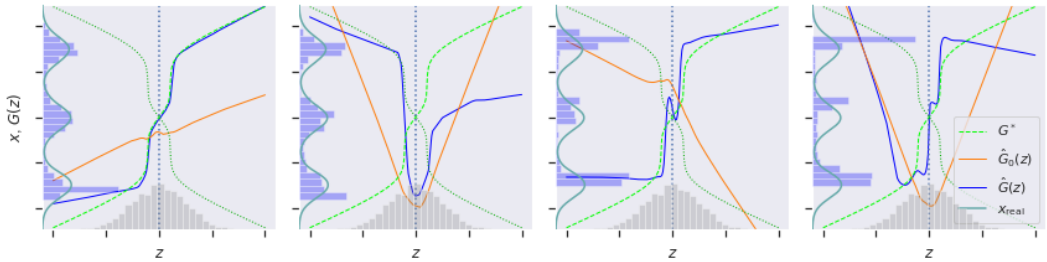


Figure 1: Initial and final generator functions trained by SGD. **Left** to **Right:** $H = 128$, He initialization; $H = 128$, V-shape initialization; $H = 2048$, He initialization; $H = 2048$, V-shape initialization. Grey and blue histograms show the distribution of latent noise and data generated by the trained generator shown in the blue curve. The orange and green curves show the initial generator and the optimal monotonic generators. Smaller GANs approximate green lines well, while larger GANs have fluctuation in the learned function (cf. Geiger et al. [2020]). V-shape init biases to polytonic generators, which have more discontinuities.

G, and 4) IB of competitive training on D. Schaefer et al. [2020] focuses on 4) and demonstrates that implicit competitive regularization (ICR) introduces additional stable points (or regions) which do not exist when only training one player while keeping the other fixed. They argue that ICR is the reason why D seems to be able to guide G, rather than exploiting the misalignment of the supports of real and fake data. We believe for G, the counterpart effect 3) is similar but harmful. ICR may prevent G from matching densities since to do so requires G to approximate discontinuities which is not suitable for NNs, which may imply that the volume of the region of solutions is relatively smaller. Then, the additional stable points may drag G away from these solutions. But in this work, we focus primarily on 1), leaving 2,3) as future directions.

Recently, Balaji et al. [2021] show that an overparameterized GAN converges fast to a global saddle point. It says less about the GAN performance since the GAN loss does not directly reflect the quality of generated data. Although their experiments show that overparameterized GANs perform better, they only used a relatively small learning rate for all settings. Empirically, we observe smaller learning rates work well for larger networks, and we think it is fairer to choose the best learning rate for each network size.

Discontinuity in the generator. It is well-known that discontinuities cause troubles in GAN training [Xiao et al., 2018, Hoang et al., 2018, Tanielian et al., 2020, An et al., 2020a], but there is few discussions on other properties of the optimal generator function. Specifically, the polytonicity (Def. 1) in the optimal generators has not been addressed. We argue that the discontinuities in optimal monotonic generators already pose a difficulty for a NN, the issue only gets worse if there are more than needed. Hence, in the next section, we provide a characterization for this.

3 Characterization of Optimal Polytonic Generators for a 1D Target Probability Density

In this section, we show that in 1D, the optimal generator can be polytonic. All derivations are deferred to Appendix A. Let $f_V(v)$ denote the probability density function (PDF) for a random variable V and let $F_V(v)$ denote the corresponding cumulative distribution function (CDF). We first define the notations as follows.

Definition 1. *A function $h : \mathbb{R}^d \to \mathbb{R}$ is **monotonically increasing (decreasing)** if $x_1 \leq x_2 \implies h(x_1) \leq (\geq) h(x_2)$. We call h monotonic if it is either monotonically increasing or decreasing. A function is **bitonic** if it is composed of two monotonic function with different monotonicities whose supports intersect at only one point. Each monotonic function is called a **piece**. More generally, a function is **polytonic of order** $p \in \mathbb{Z}^+$ (or p**-tonic**) if it is composed of p connected monotonic pieces with alternating monotonicities.*

Assumption 1. *Assume that the generator G is piecewise monotonic and differentiable, with P pieces $\{G_p : p \in [P]\}$, each with domain $\mathcal{Z}_p$, range $\mathcal{X}_p$ and monotonicity $s_p = +1(-1)$ if increasing*

(decreasing). Then, $G(z) := G_p(z)$, if $z \in \mathcal{Z}_p$. Adjacent pieces G_p, G_q, $\forall p, q \in [P]$, $|p - q| = 1$ have alternating monotonicities, i.e. $s_p \cdot s_q = -1$.

Note that each piece of h corresponds to a branch of h^{-1} [Trench, 2013], which refers to each portion of h^{-1} if it is multivalued. Next, In Thm. 1, we present the set of constraints that each monotonic piece of an optimal generator should satisfy.

Theorem 1. *Under Assumption 1, if $X = G(Z) \sim f_X$, where $Z \sim \mathcal{N}(0, 1)$ implies that G_p must satisfy the constraints*

$$\sum_{p \in \mathcal{P}(x)} \frac{s_p \phi(G_p^{-1}(x))}{\partial_u G_p(z(x))} = f_X(x)\,, \quad \forall x \in \mathcal{X} \qquad \textit{(Density Matching)} \tag{1a}$$

$$s_p \cdot \partial_z G_p(z) \geq 0\,,\ s_p = (-1)^{p-1} \cdot s_1\,, \quad \forall p \in [P] \qquad \textit{(Piecewise Monotonicity)} \tag{1b}$$

*where $\mathcal{P}(x) := \{p : \mathcal{X}_p \cap \{x\} \neq \emptyset, p \in [P]\}$ is the set of **active** pieces. $\phi(\cdot)$ is the PDF of standard Gaussian distribution.*

Polytonic generators in 1D are more constrained in low-density regions. We find that given these constraints, a counterintuitive consequence is that polytonic generators, which seem to have far more degrees of freedom than monotonic ones, are more constrained in low-density regions of $f_X(x)$ than their monotonic counterparts. We can see this from Eq. (1a) by lower bounding $G_p(z(x))$ as $\partial_z G_p(z(x))/s_p = |\partial_z G_p(z(x))| \geq \phi(G_p^{-1}(x))/f_X(x)$, $\forall p \in \mathcal{P}(x)$. In the zero-density limit $f_X(x) \to 0$, unless $\phi(G_p^{-1}(x)) \to 0$, the only solution is $|\partial_u G_p(z(x))| = \infty\,, \forall p \in \mathcal{P}(x)$, implying that *low-density regions in f_X force **all** contributing pieces of a polytonic generator to have infinitely large gradient norms*. Geometrically, this corresponds to that a zero-slope plateau in $G_p^{-1}(x)$ force any contributing piece $G_p(z(x))$, $p \in \mathcal{P}(x)$ to be discontinuous, unless there is no latent noise, which justifies the usage of latent noise distributed as mixture of Gaussians [Xiao et al., 2018]. When using NNs, whose implicit bias favors continuous functions, these regions can pose significant difficulty, greater perhaps than with an appropriately constrained monotonic generator. Note that pieces inactive in low-density regions may cause local fluctuations in the function learned, as shown in Fig. 1.

In Fig. 1, we visualize GANs trained on a 1D mixture of 3 Gaussians with SGD. With a conventional He initialization, we observe that a smaller-width generator may land on one of the optimal monotonic generators, larger-width generators have fluctuations (cf. Geiger et al. [2020]). If we intentionally bias G to bitonic with a V-shape initialization, indeed, more discontinuities need to be approximated. We also confirm the existence of polytonicity in G in Fig. 6a (see the upper right panel, G_2, the 2nd component of G).

4 Implicit Bias in the Context of GANs

In this section, we first introduce a reparameterization of the parameters of a shallow (2-layer) NN, which will help our analyses later. Then, we propose our hypotheses on the impact of IB in the context of GANs, based on our understanding from the supervised learning setting.

4.1 Breakplane Parameterization: ReLU Networks as Continuous Piecewise Linear Splines

Definition 2. *Consider a shallow ReLU NN parameterized by $w_i \in \mathbb{R}^d$, $b_i \in \mathbb{R}$ and $v_i \in \mathbb{R}^D$, whose input $x \in \mathbb{R}^d$ and output $h(x) \in \mathbb{R}^D$. For each neuron i, given NN parameters $\theta_i = (w_i, b_i, v_i)$, the **spline** parameters $\psi_i = (\xi_i, \beta_i, \mu_i)$ are defined as: $\xi_i := w_i/||w_i||_2$ is the **breakplane**'s orientation, $\beta_i := -b_i/||w_i||_2$ is the breakplane's distance from the origin, and $\mu_i := v_i||w_i||_2$ is the **delta-slope**. Then, $h(x)$ can be written as*

$$y = h(x; \theta) := \sum_{i=1}^{H} v_i \mathrm{ReLU}(w_i^T x + b_i) := \sum_{i=1}^{H} \mu_i \mathrm{ReLU}(\xi_i^T x - \beta_i), \tag{2}$$

Although the mathematical form of the spline parameterization looks similar to the original one, it has a clear geometric interpretation: 1) Each breakplane corresponds to a hyperplane that partitions the domain of input. Inputs in one half space activate the neuron while those in the other do not. 2) ξ_i

points the normal direction of the corresponding breakplane. 3) β_i is the distance to the breakplane from origin. And 4) μ_i is the contribution of neuron i to $h(x)$ when x increases one unit along direction ξ_i in the active half space of i. Note that if $D > 1$, μ_i contains the neurons contribution in each output dimension $\mu_{i,j}, \forall j \in [D]$. All dimensions share the same breakplanes. See Fig. 2 for an illustration in 1D in which case breakplanes become breakpoints and μ_i corresponds to the change of slope of h in the half space (left or right of the breakpoint) where neuron i is active.

Breakplane alignment. This spline parameterization is shown to be useful in understanding the initialization, learning dynamics, and implicit regularization for NNs in function space [Steinwart, 2019, Sahs et al., 2020]. Relating to our focus on discontinuities, it is shown in Sahs et al. [2020] that a single neuron rarely has a very large delta-slope. Hence, to implement a discontinuity in h, it typically requires many breakplanes to align with each other, each contributing some delta-slopes a little, resulting in a large $|\nabla_x h|$. In the lower left and upper right panels of Fig, 6a, breakplanes are shown in colored lines. We observe each sharp transition in the function (flat to steep, steep to flat) corresponds to a set of breakplanes aligned.

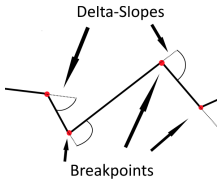


Figure 2: Breakplanes (Breakpoints) in 1D.

4.2 Hypothesis on the Impact of Kernel Regime

We hypothesize that if G is in the kernel regime, it will have a hard time approximating the discontinuities need for learning data distribution, while it will be more successful when in the adaptive regime. The reason is that when in the kernel regime, NN parameters stay around their initialization and NTK is fixed, while in the adaptive regime, they change fast and the neural tangent feature aligns quickly [Baratin et al., 2021]. From a geometric perspective (Sec. 4.1), in the kernel regime breakplane mobility is dramatically reduced, rendering it difficult to concentrate curvature in desired directions and approximate discontinuities. This will lead to poor approximations, analogous to the 'ringing' phenomenon seen when Fourier Series are used to approximate step functions [Hewitt and Hewitt, 1979]. However, in the adaptive regime, breakplanes are mobile and can align in important directions and approximate discontinuities with ease, which should improve mode coverage. Since Woodworth et al. [2020], Geiger et al. [2020], Moroshko et al. [2020] have shown that overparameterization and large scale of initialization both shift NN to kernel regime in the supervised learning setting, we can summarize that increasing width or scale of initialization causes poor GAN performance.

It is worth to mention that Moroshko et al. [2020] show that even with large initialization, NNs can eventually reach the adaptive regime if trained infinitely long. However, this is impractical. Hence, choosing the right bias for a given problem can reduce the learning time and resources needed.

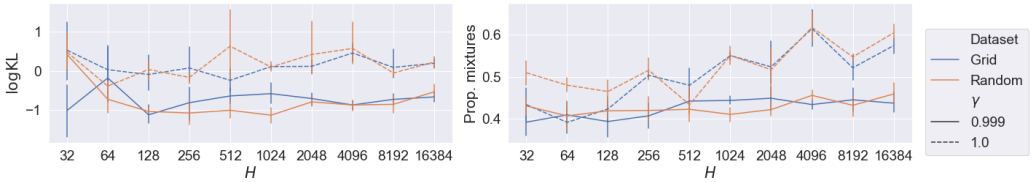
In Section 5, we test our hypothesis by training GANs in the spectrum from kernel to adaptive regime. Since adaptive optimizers like RMSProp [Tieleman and Hinton, 2012] also allow NN parameters to change faster, we interpolate from it to SGD as well.

5 Experimental Results

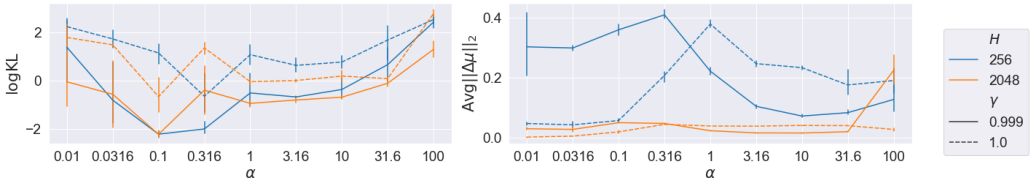
In this section, we first show results of experiments varying hidden-layer width, adaptivity of the optimizer, and scale of initialization using Shallow ReLU NNs on synthetic datasets. We follow the common practice in modern GANs like BigGAN [Brock et al., 2018] of maintaining a balance of capacity between two players, varying the hyperparameters for both networks simultaneously. Then, we apply causal analysis on the experiment data we collected, which include hyperparameters, initial and final metrics of interest, and performance metrics, to identify the causal impact of implicit bias on GAN performance. Finally, we show experiments on real-world datasets and observe similar effects. Models, datasets, and training settings used are shown in Appendix C.

5.1 Shallow ReLU GANs with Mixture of Gaussian Datasets

We show GAN performance for each combination of H and γ in Fig. 3a, and α and γ in Fig. 3b. Learning rates are tuned from $\{10^{-6}, 10^{-5.5}, \cdots, 10^{-3.5}, 10^{-3}\}$. In Fig. 3a, we observe GAN performance is the best with smaller width (512 for SGD and 128 for RMSProp), as long as the NNs have sufficient expressive power. Since for each H, we choose the best learning rate, we conclude



(a) $\alpha = 1$. **Left:** final performance. **Right:** proportion of mode mixtures in generated data.



(b) Grid dataset. **Left:** final performance, $H = 256$. **Right:** norm of delta-slope absolute updates.

Figure 3: **Top:** GAN performance and proportion of mode mixtures using different widths H , RMSProp parameter γ, dataset, and the corresponding best learning rate. Error bars are calculated across 5 runs. **Bottom:** GAN performance and the average of norms of absolute updates of delta-slopes (Def. 2) using different H, γ, α, and the corresponding best learning rate. We observe overparameterization, large scale of initialization, and non-adaptive optimizer hurt GAN performance. Overparameterization or SGD leads to more mode mixtures. Networks starting with small initialization have an even larger norm of *absolute* updates in delta-slopes. Larger networks or SGD leads to smaller average updates.

that overparameterization actually *hurts* GAN performance by biasing the NNs to kernel regime. While this might seem opposite to previous work [Balaji et al., 2021] where they use a fixed learning rate, smaller networks do perform better with tuned learning rates. This can also be confirmed in Fig. 3a, where larger networks have more mode mixtures.

Fig. 3b shows that a moderately small scale of initialization also improves GAN performance in both SGD and RMSProp cases, achieving even lower KL than any cases with $\alpha = 1$. This holds for both $H = 256$ and $H = 2048$. As we know, larger widths or scales tend to linearize the networks and shift them into the kernel regime, it implies there must be some properties of a GAN in the kernel regime that leads to mode collapse or mixture. Fig. 3b shows the update of G's delta-slopes (see Section 4.1) after training. We observe NNs starting with small initialization have even larger updates than those with large initialization, which confirms that they are in different regimes.

In addition, there is a clear difference in the trend between $\gamma < 1$ (even if γ is very close to 1) and $\gamma = 1$ (Appendix B), which correspond to RMSProp with different parameters and SGD. This confirms the argument in Liu et al. [2019] that adaptivity in the optimizer help improve GAN performance. This also justifies the common practice that people just use the default parameter of adaptive optimizers.

We also run experiments, in which we only modify G, since it directly controls the quality of generated samples and we can exclude the possible impact from a varying discriminator. In this case, G still has better performance when $H_G = 128$ is small or $\alpha_G = 0.316$ is small. The figures are shown in Appendix B.

5.2 Identify Causal Effect of Implicit Bias on GAN Performance

In order to explore the specific role of IB on GAN performance, we attempt to estimate the *causal* treatment effect of several metrics that serve as proxies of IB. However, we have no direct control over IB itself, but only the hyperparameters, which may change IB metrics and optimization metrics simultaneously. Due to this reason, we may get biased estimates of effects for the IB metrics. Thus, we make use of *instrumental variables* (IV), which can reduce this bias [Peters et al., 2017].

Setup. In Fig. 4, we posit a high-level causal graphical model based on our knowledge on how GANs are trained and evaluated that describes the causal links between the components involved. The hyperparameters define the overall architecture of the GAN, hence they can potentially influence the initial parameters, the optimizer, and IB metrics (eg., metrics related to width). Given initial

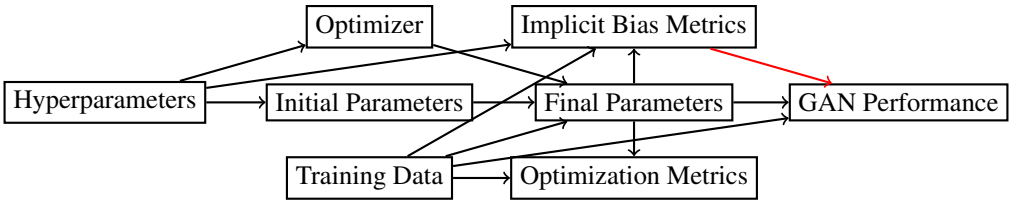


Figure 4: Causal graph of GAN training. Each arrow represents a potential causal link between components of the GAN or the data, and the red arrow represents the causal link of interest.

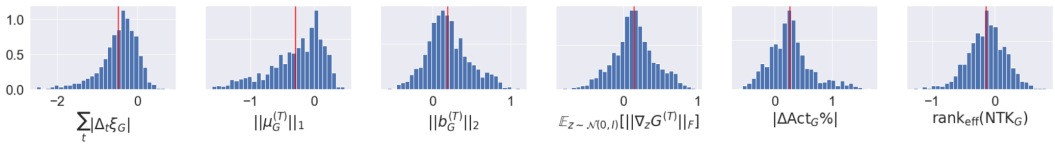


Figure 5: Distribution of marginal treatment effects evaluated at each final treatment value after normalization. Each mean effect (red) shows an overall direction. We observe that more BP reorientation (1st column), larger delta-slopes (2nd) and faster alignment of neural tangent features (6th) cause G to better learn the data distribution, while larger bias norm (3th), larger Jacobian Frobenius norm over latent noise (4th), and more activation change (5th) cause G to perform worse.

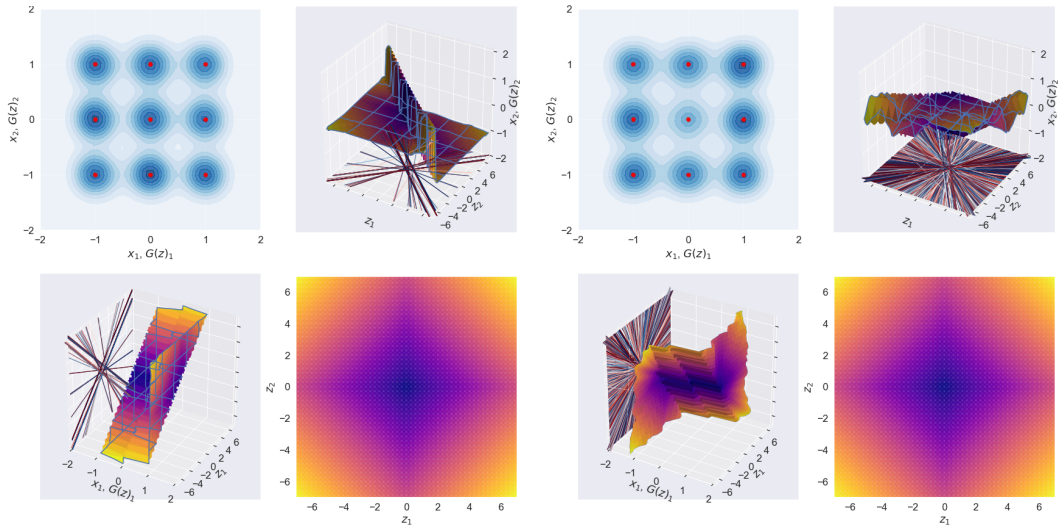
parameters, an optimizer, and a training dataset, we obtain the final parameters, which determine the implicit bias metrics, optimization metrics, and overall performance of the GAN.

We estimate a non-linear IV model in which the initial parameters are instruments, the implicit bias metrics are the treatment variables, the final parameters and optimization metrics are included as control variables, and the performance metric is the outcome of interest. Specifically, we use 4-layer fully connected networks for both the treatment model and response model in DeepIV [Hartford et al., 2017] implemented by EconML [Research, 2019]. Provided that the initial G parameters affect GAN performance only through these metrics and the final G parameters, this model will yield the causal effects of interest.

To implement this model, we use the GAN loss as the optimization metric and KL divergence as the performance metric. Our implicit bias metrics include Jacobian Frobenius norms (JFNs) $||\nabla_z G(z)||_F$ of G w.r.t. latent z, averaged over $\mathcal{P}_z$, norms of weights and their updates after training, spline parameters (Def. 2) and the change of activation pattern $|\Delta \mathrm{Act}_G\%|$, and the effective rank of the neural tangent kernels (NTKs) $\mathrm{rank}_{\mathrm{eff}}(\mathrm{NTK})$ [Baratin et al., 2021]. Since JFNs are large at low-quality generated samples [Tanielian et al., 2020], they can be used for rejection sampling and measuring the roughness of G. The norms of weights and their updates are essential for many analyses on overparameterization to be applicable since they require linearizing the network, or the learned weights are close to their initial values [Li and Liang, 2018, Jacot et al., 2018, Chizat et al., 2019]. The spline parameters (Def. 2) characterize the function geometry, and $|\Delta \mathrm{Act}_G\%|$ can be used to determine if the network is in the kernel regime [Li and Liang, 2018]. Finally, as it has been shown that implicit regularization induces a dynamic alignment of neural tangent features that helps feature selection and compression in classification tasks, we include $\mathrm{rank}_{\mathrm{eff}}(\mathrm{NTK})$ to gauge whether this alignment corresponds to the BP alignment.

Results. The two-stage model has an R^2 of 0.7102, indicating it has strong predictive power. We monitor the training and validation loss throughout training and find they stay close to each other. It is necessary to rule out the possibility of overfitting since NNs can even fit random noise [Zhang et al., 2017]. We compute the marginal treatment effects (MTEs) around observed treatment values conditional on the control variables. In Fig. 5, we present the distributions of MTEs.

We observe negative average MTEs of the breakplane (BP) orientation updates, delta-slope norms, and effect rank of NTK of G, which means that higher values of these variables improve GAN performance. This is consistent with our predictions: i) More orientation updates lead to more BP alignment, ii) large delta-slope norm, i.e. more curvature induced by the breakplanes, make G able to approximate discontinuities, iii) the effect rank of NTK itself measures alignment of neural tangent



(a) $H = 128$, log KL $= -2.197$ (b) $H = 2048$, log KL $= -1.371$

Figure 6: G learned for 2D Grid mixture of Gaussian data. For each width: **Upper Left:** Kernel density estimates of generated data (blue) and real data samples (red). **Lower left** and **Upper right:** The 1st and 2nd dimension G_1, G_2 of $G(z)$ with breakplanes color-weighted by delta-slopes. For G_1, the output is shown in the horizontal axis to match the upper left panel. **Lower right**: Color mapping from z to $G(z)$. A smaller network learns a much simpler G function whose most regions are very smooth and have discontinuities only in low-density regions, while the larger GAN mismatches the mode frequencies and has fluctuations in the learned function.

features. With less rank, the features become more correlated and able to express a steeper function after combined by the final layer.

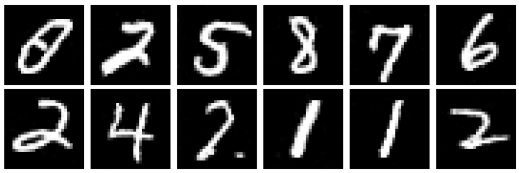
Next, we find that larger bias norms, JFNs over latent noise, and more activation change $|\Delta\mathrm{Act}_G\%|$ cause G to perform worse. Note that there is an interesting contrast between the norm of the hidden layer bias and the norm of delta-slopes, which is the product of hidden layer weight and output layer weight. This implies different groups of parameters play different roles and we may need different adaptivity. Intuitively, when bias is greater, there is less relative change in the outputs for the same size change in the inputs. This hurts G, since we would expect it to generate samples at different modes instead of mode mixtures even when we smoothly interpolate the latent space.

While the effect of JFNs is consistent with Tanielian et al. [2020], the effect of $|\Delta\mathrm{Act}_G\%|$ is positive, meaning it causes the model to perform worse. This is surprising, as we typically expect NNs with high activation pattern change to be in the adaptive (rich) regime. However, it does not directly measure if the breakplanes align with each other to approximate discontinuities. It only measures how many times they sweep across a data point, which may depend more on the limiting cycles observed in adversarial settings like GANs [Schaefer et al., 2020, Wang et al., 2019].

Finally, we visualize the actual learned Gr function of $H = 128$ and $H = 2048$ with $\gamma = 0.9$ and the best learning rate in Fig. 6. In the smaller network (lower-left and upper-right panels), the breakplanes are well aligned, which results in a simple Gr function with discontinuities only in low-density regions. In the larger network (upper-left and lower-right panels), the breakplanes remain around their initial locations (note their initial orientation is uniformly distributed, which can be seen in Sahs et al. [2020] and Appendix B) and the delta-slopes are also reluctant to change, which leaves local fluctuations everywhere in the function. Although both networks cover all the modes, the mode frequencies in the larger network are mismatched. G_2 in the $H = 128$ case is analogous to a 'polytonic' function: starting from the third quadrant of the z plane to the first, the function value increases then decreases.

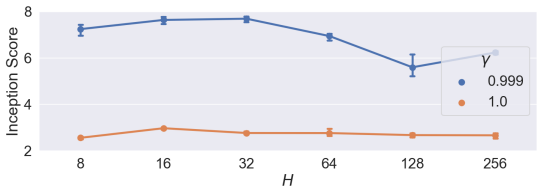
5.3 Real-World Datasets

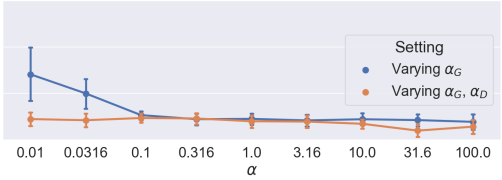
In Fig. 7, we show results of training DCGANs on MNIST [LeCun et al., 1998]. We observe in Fig. 7c that the best model with $\alpha_{G,D} = 1$ has $H = 32$, $\gamma = 0.999$, while the best model with $\alpha \neq 1$



(a) Generated digits. $H = 32$, $\gamma = 0.999$, $\alpha_G = 1$, $\alpha_D = 1$, IS $= 7.7967$

(b) Generated digits. $H = 32$, $\gamma = 1$, $\alpha_G = 0.01$, $\alpha_D = 1$, IS $= 6.5527$





(c) **Left:** Inception scores with different H and γ ($\alpha = 1$), tuned by learning rates. Error bars are calculated for 3 runs. **Right:** Inception scores with different α and setting ($H = 32$, $\gamma = 1$), tuned by learning rates.

Figure 7: G needs to be adaptive to achieve a high inception score on generating images. This can be achieved by either i) using a smaller network ($H = 32$), ii) using a more adaptive optimizer ($\gamma = 0.999$), or iii) using a smaller scale of initialization ($\alpha_G = 0.01$).

has $\alpha_G = 0.01$ and $\alpha_D = 1$. In Appendix B, we also show that the norm of parameter update in these models are the largest, indicating stronger adaptivity. Hence, this observation is consistent with what we have for the mixture of Gaussian experiments: i) If the networks are too large, ii) if the optimizer lacks adaptivity, or iii) if the scale of initialization of the network is large, G will not be able to learn a complicated multimodal data distribution. By simply turning down α_G, we obtain a decent improvement in the generated images (Fig. 7b) while G with large α_G produce basically noise images only. The fact that smaller α_D hurts may indicate that it should not learn too fast, otherwise G faces a vanishing gradient issue. Then, no matter how adaptive G is, it will still have mode collapse. We argue that this issue is more severe in real-world datasets since the tasks are more difficult, which might explain why we do not observe this in synthetic datasets.

6 Conclusion & Broader Impact

In this work, we are motivated to get a fundamental understanding of the difficulty in the task for the generator (G). In this regard, we first characterize the polytonicity in the optimal G and find that allowing polytonicity in G is seemingly more flexible but requires more discontinuities, especially in low data density regions. Then, We argue that overparameterization, large scale of initialization, or lack of adaptivity shifts G into the kernel regime and prevents it from approximating discontinuities and causes bad GAN performance. Then, we support this with causal analysis in synthetic datasets and verify it empirically in MNIST. Our findings can help with hyperparameter search for GANs since now we know that smaller networks have a more proper IB. Instead of model widths, We can save the computational budget for other hyperparameters like learning rates.

One limitation in this work is that the geometric view using breakplanes needs to be generalized to deep networks, in which case breakplanes are not hyperplanes anymore but hypersurfaces [Lee et al., 2009]. Although our theory on polytonicity only works on 1D, it is already inspiring us for a possible way to improve GANs. Generalizing our theory to higher dimensions will help us enforce monotonicity in G, which is likely to improve GAN performance. And a natural future direction of this work is to analyze the causal impact of IB on the discriminator and whether it has an IB different from NNs in supervised learning since it also needs to guide G besides the classification.

Since the aim of this work is to provide insights that can promote further improvement on GAN performance, we do not see any potential negative impacts of our work.

References

Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*,

2017.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

Yuanhao Wang, Guodong Zhang, and Jimmy Ba. On solving minimax optimization locally: A follow-the-ridge approach. *arXiv preprint arXiv:1910.07512*, 2019.

Cédric Villani. *Optimal transport: old and new*, volume 338. Springer, 2009.

Dongsheng An, Yang Guo, Min Zhang, Xin Qi, Na Lei, and Xianfang Gu. Ae-ot-gan: Training gans from data specific latent distribution. In *European Conference on Computer Vision*, pages 548–564. Springer, 2020a.

Chang Xiao, Peilin Zhong, and Changxi Zheng. Bourgan: Generative networks with metric embeddings. In *Advances in neural information processing systems*, 2018.

Quan Hoang, Tu Dinh Nguyen, Trung Le, and Dinh Phung. Mgan: Training generative adversarial nets with multiple generators. In *International conference on learning representations*, 2018.

Dongsheng An, Yang Guo, Na Lei, Zhongxuan Luo, Shing-Tung Yau, and Xianfeng Gu. Ae-ot: A new generative model based on extended semi-discrete optimal transport. In *International Conference on Learning Representations*, 2020b. URL `https://openreview.net/forum?id=HkldyTNYwH`.

Florian Schaefer, Hongkai Zheng, and Animashree Anandkumar. Implicit competitive regularization in gans. In *International Conference on Machine Learning*, pages 8533–8544. PMLR, 2020.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018.

Suriya Gunasekar, Jason D Lee, Daniel Soudry, and Nati Srebro. Implicit bias of gradient descent on linear convolutional networks. In *Advances in Neural Information Processing Systems*, pages 9461–9471, 2018.

Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. In *Advances in Neural Information Processing Systems*, pages 2937–2947, 2019.

Justin Sahs, Ryan Pyle, Aneel Damaraju, Josue Ortega Caro, Onur Tavaslioglu, Andy Lu, and Ankit Patel. Shallow univariate relu networks as splines: Initialization, loss surface, hessian, & gradient flow dynamics. 2020.

Blake Woodworth, Suriya Gunasekar, Pedro Savarese, Edward Moroshko, Itay Golan, Jason Lee, Daniel Soudry, and Nathan Srebro. Kernel and rich regimes in overparametrized models. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=Byg9bxrtwS`.

Edward Moroshko, Blake E Woodworth, Suriya Gunasekar, Jason D Lee, Nati Srebro, and Daniel Soudry. Implicit bias in deep linear classification: Initialization scale vs training accuracy. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 22182–22193. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/fc2022c89b61c76bbef978f1370660bf-Paper.pdf`.

Roman Novak, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A Alemi, Jascha Sohl-Dickstein, and Samuel S Schoenholz. Neural tangents: Fast and easy infinite neural networks in python. In *International Conference on Learning Representations*, 2019.

Josue Ortega Caro, Yilong Ju, Fabio Anselmi, Sourav Dey, Ryan Pyle, and Ankit Patel. Using learning dynamics to explore the role of implicit regularization in adversarial examples. *arXiv preprint arXiv:2006.11440*, 2020.

Mario Geiger, Stefano Spigler, Arthur Jacot, and Matthieu Wyart. Disentangling feature and lazy training in deep neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2020 (11):113301, 2020.

Aristide Baratin, Thomas George, César Laurent, R Devon Hjelm, Guillaume Lajoie, Pascal Vincent, and Simon Lacoste-Julien. Implicit regularization via neural feature alignment. In *International Conference on Artificial Intelligence and Statistics*, pages 2269–2277. PMLR, 2021.

Jaehoon Lee, Lechao Xiao, Samuel S Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in neural information processing systems*, 2019.

Yogesh Balaji, Mohammadmahdi Sajedi, Neha Mukund Kalibhat, Mucong Ding, Dominik Stöger, Mahdi Soltanolkotabi, and Soheil Feizi. Understanding over-parameterization in generative adversarial networks. In *Proceedings of the international conference on learning representations*, 2021.

Ugo Tanielian, Thibaut Issenhuth, Elvis Dohmatob, and Jérémie Mary. Learning disconnected manifolds: a no gan's land. In *International Conference on Machine Learning*, pages 9418–9427. PMLR, 2020.

William F Trench. Introduction to real analysis. 2013.

Ingo Steinwart. A sober look at neural network initializations. *arXiv preprint arXiv:1903.11482*, 2019.

Edwin Hewitt and Robert E Hewitt. The gibbs-wilbraham phenomenon: an episode in fourier analysis. *Archive for history of Exact Sciences*, 21(2):129–160, 1979.

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

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2018.

Mingrui Liu, Youssef Mroueh, Jerret Ross, Wei Zhang, Xiaodong Cui, Payel Das, and Tianbao Yang. Towards better understanding of adaptive gradient algorithms in generative adversarial nets. *arXiv preprint arXiv:1912.11940*, 2019.

Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

Jason Hartford, Greg Lewis, Kevin Leyton-Brown, and Matt Taddy. Deep iv: A flexible approach for counterfactual prediction. In *International Conference on Machine Learning*, pages 1414–1423. PMLR, 2017.

Microsoft Research. EconML: A Python Package for ML-Based Heterogeneous Treatment Effects Estimation. https://github.com/microsoft/EconML, 2019. Version 0.11.

Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Advances in Neural Information Processing Systems*, pages 8157–8166, 2018.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *Proceedings of the international conference on learning representations*, 2017.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Jeffrey M Lee, Bennett Chow, Sun-Chin Chu, David Glickenstein, Christine Guenther, James Isenberg, Tom Ivey, Dan Knopf, Peng Lu, Feng Luo, et al. Manifolds and differential geometry. *Topology*, 643:658, 2009.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] We state clear what the contributions and scope are. See Section 1.
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 6.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] This paper is not about applications. The datasets we used are either synthetic or nonprivate.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 3.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix A.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Appendix C.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix C.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Fig. 3a 3b 7c.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix C.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 5.
   (b) Did you mention the license of the assets? [Yes] See Appendix C.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No] We are not releasing new assets.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] We use public datasets.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] This is not applicable to the dataset we use.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [No] We are not doing so.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [No]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [No]