HAL
open science

# Unveiling the Latent Space Geometry of Push-Forward Generative Models

Thibaut Issenhuth, Ugo Tanielian, Jérémie Mary, David Picard

**To cite this version:**

Thibaut Issenhuth, Ugo Tanielian, Jérémie Mary, David Picard. Unveiling the Latent Space Geometry of Push-Forward Generative Models. Fortieth International Conference on Machine Learning, Jul 2023, Hawaii Waikola, United States. hal-04443284

**HAL Id: hal-04443284**

**https://hal.science/hal-04443284v1**

Submitted on 7 Feb 2024

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Unveiling the Latent Space Geometry of Push-Forward Generative Models

**Thibaut Issenhuth** [1 2] **Ugo Tanielian** [1] **Jérémie Mary** [1] **David Picard** [2]

## Abstract

Many deep generative models are defined as a push-forward of a Gaussian measure by a continuous generator, such as Generative Adversarial Networks (GANs) or Variational Auto-Encoders (VAEs). This work explores the latent space of such deep generative models. A key issue with these models is their tendency to output samples outside of the support of the target distribution when learning disconnected distributions. We investigate the relationship between the performance of these models and the geometry of their latent space. Building on recent developments in geometric measure theory, we prove a sufficient condition for optimality in the case where the dimension of the latent space is larger than the number of modes. Through experiments on GANs, we demonstrate the validity of our theoretical results and gain new insights into the latent space geometry of these models. Additionally, we propose a truncation method that enforces a simplicial cluster structure in the latent space and improves the performance of GANs.

## 1. Introduction

GANs (Goodfellow et al., 2014) and VAEs (Kingma and Welling, 2014) have shown great capacities to generate photorealistic images (Karras et al., 2021; Vahdat and Kautz, 2020). These two models are also helpful for diverse tasks such as image editing (Shen et al., 2020; Wu et al., 2021) or unsupervised image segmentation (Abdal et al., 2021; Zoran et al., 2021). GANs and VAEs rely on learning a Lipschitz-continuous transformation from a low dimensional Gaussian space. As such, they have been described as *push-forward generative models* (Salmona et al., 2022). According to the same taxonomy, score-based models can be defined as *indirect push-forward generative models* since they result from the composition of a large number of transformations and are trained with an auxiliary denoising objective.

The present paper aims at making a step towards a better understanding of push-forward generative models such as GANs. In particular, the goal is to shed light on the latent space of these architectures, and to stress how it impacts the performance of both GANs and VAEs. If empirical studies such as Donahue and Simonyan (2019) have suggested the emergence of simple geometrical structure in the latent space of GANs, there is still a poor theoretical understanding of how generators organize their latent space. We would like to highlight that although our theoretical results apply to all pushforward generative models, our primary emphasis in the experiment section is on GANs.

To better understand the latent space of generative models, the setting of disconnected distributions learning is enlightening. Experimental and theoretical works (Khayatkhoei et al., 2018; Tanielian et al., 2020; Salmona et al., 2022) have shown a fundamental limitation of push-forward generative models. Since the modeled distribution is connected, some areas of its support are necessarily mapped outside the true data distribution. However, when covering several modes of a disconnected distribution, generators still try to minimize the numbers of samples lying outside the true modes (*e.g.* the purple area on the right of Figure 1). In other words, generators aim at minimizing the measure of the existing borders between the modes in the latent space. Considering a Gaussian latent space, finding such minimizers is closely linked to Gaussian isoperimetric inequalities (Ledoux, 1996) where the goal is to derive the partitions that split a Gaussian space with minimal Gaussian-weighted perimeters. Most notably, a recent result (Milman and Neeman, 2022) shows that, as long as the number of components $m$ in the partition and the number of dimensions $d$ of the Gaussian space are such that $m \leq d + 1$, the optimal partition is a 'simplicial cluster': a Voronoi diagram with equidistant seeds, see left of Figure 1 for $m = 3$ and $d = 3$.

In this paper, we demonstrate the effectiveness of applying simplicial clusters to the latent space of push-forward generative models. We show both experimentally and theoretically that generators with a latent space structured as a simplicial cluster minimize the occurrence of out-of-distribution generated samples. Using the *precision* metric (Sajjadi et al.,

[1]Criteo AI Lab, Paris, France [2]LIGM, Ecole des Ponts, Univ Gustave Eiffel, CNRS, Marne-la-Vallée, France. Correspondence to: Thibaut Issenhuth <thibaut.issenhuth@live.fr>.

*Proceedings of the $40^{th}$ International Conference on Machine Learning*, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

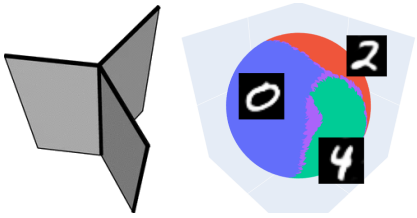


*Figure 1.* Illustration of the capability of GANs to discover an optimal geometry of the latent space. On the left, the propeller shape represents a partition of 3D Gaussian space with the smallest Gaussian-weighted perimeter (Figure from (Heilman et al., 2013)). On the right, we show the 3D Gaussian latent space of a GAN trained on three classes of MNIST. Each area colored in blue, green, or red corresponds to samples in one of the three classes. Using a pre-trained classifier, we highlight in purple the samples with low-confidence, and observe that the partition reached by the GAN (right) is close to optimality (left), as the latent space partition is similar to the intersection of the propeller on a sphere.

2018; Kynkäänniemi et al., 2019), we show that generators with a simplicial cluster latent space achieve optimal precision levels and provide both an upper and a lower bound on their precision. Our experiments reveal that GANs with higher performances tend to organize their latent space as simplicial clusters. More importantly, we illustrate that enforcing this 'simplicial structure' with a truncation method can boost GANs' performance. Interestingly, simplicial clusters are highly similar to the 'simplex Equiangular Tight Frames' observed in the last-layer features of deep classification networks (Papyan et al., 2020). This study stresses that they also naturally emerge in deep push-forward generative models. Our contributions are the following:

- We are the first to build on the latest results from Gaussian isoperimetric inequalities by Milman and Neeman (2022) in the study and understanding of push-forward generative models.
- We present a new theoretical analysis, providing both an upper bound on the precision of push-forward generative models. We demonstrate that generators with a latent space organized as a simplicial cluster have an optimal precision, with lower bounds that decrease in $\sqrt{m \log m}$, where $m$ is the number of modes.
- Experimentally, we verify that GANs tend to structure their latent space as simplicial clusters' by exploring two properties of the latent space: linear separability and convexity of classes. Also, we analyse the impact of latent space dimension on GANs, and reveal a positive correlation between GANs' performance and latent space geometry.
- Finally, we show that enforcing a simplicial structure into GANs' latent space can boost their performance and outperforms other boosting methods.

## 2. Related Work

### 2.1. Notation

**Data.** We consider a target distribution $\mu_\star$ defined on a Euclidean space $\mathbb{R}^D$, which may be a high-dimensional space, and equipped with the Euclidean norm $\|\cdot\|$. We use $S_\mu$ to represent the support of any distribution $\mu$.

**Push-forward generative models.** We consider the set of $L$-Lipschitz continuous functions, denoted as $\mathscr{G}_L$, from the latent space $\mathbb{R}^d$ to the high-dimensional space $\mathbb{R}^D$. The primary goal of each generator in this set is to produce realistic samples. The distribution in the latent space, defined on $\mathbb{R}^d$, is assumed to be Gaussian and is represented as $\gamma$. For each generator $G \in \mathscr{G}_L$, we associate the push-forward distribution (or image distribution) of $\gamma$ by $G$, and denote it $G\sharp\gamma$, where $\sharp$ denotes the push-forward operator. In the context of generative models, each distribution $G\sharp\gamma$ is now a candidate distribution to represent $\mu_\star$.

The Lipschitzness assumption on $\mathscr{G}_L$ is reasonnable: Virmaux and Scaman (2018) have shown the lipschitzness of deep neural networks, and have developed an algorithm that can upper-bound their Lipschitz constant. While deep neural networks can have high Lipschitz constants, it is possible to constrain this in practice by techniques such as clipping the neural network's parameters (Arjovsky et al., 2017), penalizing the discriminative functions' gradient (Gulrajani et al., 2017; Kodali et al., 2017; Wei et al., 2018; Zhou et al., 2019), or penalizing the spectral norms (Miyato et al., 2018). Large-scale generators such as SAGAN (Zhang et al., 2019) and BigGAN (Brock et al., 2019) also make use of spectral normalization for the generator.

### 2.2. Generative models and disconnected distributions

The phenomenon of misspecification in continuous generative models, while primarily studied in the context of GANs, is also relevant to other families such as VAEs or normalizing flows (Salmona et al., 2022). This issue has been investigated both experimentally (Khayatkhoei et al., 2018) and theoretically (Tanielian et al., 2020; Salmona et al., 2022). The problem stems from a fundamental trade-off: continuous generators can either cover all modes, resulting in out-of-manifold samples, or generate only high-quality samples, neglecting some modes. To address this, various methods have been proposed, such as training disconnected distributions (Gurumurthy et al., 2017; Khayatkhoei et al., 2018) or deriving rejection mechanisms from pre-trained generators (Azadi et al., 2018; Tanielian et al., 2020; Humayun et al., 2022).

Empirical studies have provided valuable insights into the

structure of the latent space of generative models. For example, Karras et al. (2019) demonstrate that binary attributes are linearly separable in the Gaussian latent space and even more separable in an intermediate latent space. Similarly, Shen et al. (2020) find that face attributes are separated by hyperplanes in the latent space.Arvanitidis et al. (2018) and Chen et al. (2018a) view the latent space of generative models with a Riemannian perspective.

While these findings provide valuable insights into the latent space structure of generative models, they may not be sufficient for a comprehensive understanding of the latent space geometry. For instance, Tanielian et al. (2020) stress the relevance of this problem by showing that the precision of GANs can converge to 0 when the number of modes or the distance between them increases. In this paper, we take a step towards a deeper understanding of the behavior of push-forward generative models and reveal an optimal latent space configuration when the number of modes $m$ and the dimension of the latent space $d$ are such that $m \leq d+1$.

### 2.3. Evaluating generative models

When learning disconnected manifolds, Sajjadi et al. (2018) illustrated the need for measures that simultaneously evaluate both the quality (Precision), and the diversity (Recall) of the generated samples. However, Kynkäänniemi et al. (2019) pointed out an important limitation of the PR metric: it cannot accurately interpret situations when large numbers of samples are packed together. They propose an Improved PR metric based on the non-parametric estimation of manifolds to correct this.

**Improved PR metric.** Informally, for a generator $G$, precision ($\alpha_G$) quantifies the proportion of generated samples that can be approximated with true samples, while recall ($\beta_G$) measures the proportion of true samples that can be approximated with generated ones. Applying this to GANs, using the target distribution $\mu^\star$ and modeled distribution $G\sharp\gamma$, the Improved PR metric was shown, by Tanielian et al. (2020, Theorem 1), to be asymptotically equivalent to:

$$\alpha_G^n \underset{n\to\infty}{\to} \alpha_G = G\sharp\gamma\big(S_{\mu^\star}\big) \text{ and } \beta_G^n \underset{n\to\infty}{\to} \beta_G = \mu^\star\big(S_{G\sharp\gamma}\big),$$

where $S_{\mu^\star}$ denotes the support of $\mu^\star$ and $n$ is the number of samples. However, Naeem et al. (2020) have shown that the Improved PR metric (Kynkäänniemi et al., 2019) is sensitive to outlier samples of both the target and the generated distribution. To correct this and fix the overestimation of the manifold around real outliers, Naeem et al. (2020) propose the Density/Coverage metric.

**Density/Coverage.** Instead of counting how many fake samples belong to a real sample neighborhood, density counts how many real sample neighborhoods contain a generated sample. On the other hand, coverage counts the number of real sample neighborhoods that contain at least one fake sample.

In the next analysis both theoretical and experimental, we use both notions of precision and density defined above.

## 3. Simplicial Structure in Push-Forward Generative Models

The goal is to gain a deeper understanding of the latent space of push-forward generative models and identify which ones possess the highest precision under certain conditions. As previously mentioned, push-forward generative models map a unimodal Gaussian distribution $\gamma$ through a Lipschitz-continuous function, represented by a generator $G$. As a result, the modeled generative distribution $G\sharp\gamma$ necessarily has a connected support.

In cases where the target distribution $\mu^\star$ contains disconnected manifolds, generators have to generate fake data points that fall outside of the true manifold. This prompts the question: given that a generator samples data points from each of the distinct modes, what is the maximum precision that it can achieve? To begin with, let's consider a target distribution $\mu^\star$ composed of $m$ disconnected modes.

**Assumption 3.1** (Disconnected manifolds)**.** The target distribution $\mu^\star$ consists of $m$ disconnected spheres $S_i, i \in [1,m]$ of equal measure (with centers $X_i$ and radius $r_i$). Additionally, the spheres satisfy the two following properties:

- **Small individual radius:** each radius $r_i$ satisfies

$$r_i < \min_j \frac{\|X_i - X_j\|}{2}. \tag{1}$$

- Each distance $\|X_i - X_j\|$ satisfies:

$$\min_{k\in[1,m], k\neq i,j} \|(X_i+X_j)/2 - X_k\| > \frac{\|X_i - X_j\|}{2}. \tag{2}$$

We believe that the assumption of disconnectedness is a reasonable one, particularly for multi-class datasets such as MNIST (LeCun et al., 1998), CIFAR10 (Krizhevsky, 2009), or STL10 (Coates et al., 2011). To validate this property, we run a pre-trained CLIP (Radford et al., 2021) on the dataset, identify a certain number of clusters using a K-means algorithm, and further test the disconnectedness of these modes by training a linear classifier. The accuracy on these datasets is 98.1% on MNIST, 93.9% on CIFAR10, and 92.7% on STL10.

The second point in (2) has a direct impact on the location of the data points $X_1, \dots, X_m$. Specifically, it implies that each cell in the Voronoi diagram with seeds $X_1, \dots, X_m$ shares a side with all the other cells. In other words, the dual

graph of this Voronoi diagram is complete. This assumption, which is further discussed with specific examples in Figure 2, can be justified by the concentration of distances in high-dimensional spaces: all the modes are roughly at equal distance (Beyer et al., 1999; Aggarwal et al., 2001). Furthermore, a recent work by Papyan et al. (2020) has shown that embeddings of deep neural networks trained for classification tend to collapse around means that are equidistant and maximally equiangular to one another. By using these embedded representations to measure distance, the target distribution would thus easily satisfy Assumption 3.1. Projected GANs (Sauer et al., 2021) is really close to this idea as the authors show the effectiveness of leveraging a pre-trained classifier when training GANs: instead of directly discriminating images, the discriminator is trained on features extracted from the classifier.

Throughout the rest of the paper, we define the set of *well-balanced* generators as those mapping an equal number of data points to each mode of the data distribution:

**Definition 3.2.** A generator $G$ is well-balanced if for all spheres, we have $G\sharp\gamma(S_1) = \ldots = G\sharp\gamma(S_m)$.

Considering well-balanced generators is reasonable as many empirical improvements such as WGAN-GP (Gulrajani et al., 2017) or BigBiGAN (Donahue and Simonyan, 2019) have significantly reduced mode collapse. GANs generate diverse output distributions on datasets such as CIFAR10, CIFAR100, and ImageNet. To validate the use of well-balanced generators, we conducted a small experiment and evaluated the proportion of each class generated by GANs on MNIST and CIFAR10. On MNIST, the minimum proportion of a class is $9.2$ and the maximum $10.9$, while on CIFAR10 it is $8.3$ and $11.9$ (in %). The variance-to-mean ratio is equal to $0.03$ for MNIST and $0.22$ for CIFAR10.

### 3.1. Precision and the associated partition

Now that we have defined the prerequisites for both the data and the model, we propose to establish a connection between the latent space partition and the precision of a generator. We create a link between the set of generators from $\mathbb{R}^d$ to $\mathbb{R}^D$ and the set of partitions in the latent space. Specifically, for each given partition in $\mathbb{R}^d$, there exists a set of associated generators defined as follows:

**Definition 3.3.** For a given partition $\mathcal{A} = A_1, \ldots, A_m$ on $\mathbb{R}^d$, we say that $G$ is associated to $\mathcal{A}$ if: for all $i \in [1, m]$, for all $z \in A_i$, $i = \underset{j\in[1,m]}{\arg\min} \|G(z) - X_j\|$.

Each given generator $G$ is associated with a unique partition $\mathcal{A}$ in $\mathbb{R}^d$. The geometry of the associated partition $\mathcal{A}$ plays a key role in explaining the behavior and performance of the generator $G$. We are interested in maximizing the precision of generative models. Points in the intersection of two cells $A_i \cap A_j, (i, j) \in [1, m]^2$ are equidistant from $X_i$ and $X_j$ and thus do not belong to any of these modes (since bot $r_i$ and $r_j < \|X_i - X_j\|/2$ according to Assumption 3.1). Additionally, due to the generator's Lipschitzness, there is a small neighborhood around the boundary such that any points in this neighborhood are mapped out of the target manifold. This region in the latent space thus reduces the precision. For a given $\varepsilon > 0$, we now define the epsilon-boundary of the partition $\mathcal{A}$ as follows.

**Definition 3.4.** For a given partition $\mathscr{A} = \{A_1, \ldots, A_m\}$ of $\mathbb{R}^d$ and a given $\varepsilon \in \mathbb{R}_+^\star$, we denote $\partial^\varepsilon \mathcal{A}$ the $\varepsilon$-boundary of $\mathcal{A}$, defined as follows.

$$\partial^\varepsilon \mathcal{A} = \bigcup_{i=1}^{m} \big(\cup_{j\neq i} A_j\big)^\varepsilon \backslash \big(\cup_{j\neq i} A_j\big),$$

where $A^\varepsilon$ corresponds to the $\varepsilon$-extension of set $A$. The following lemma makes the connection between the precision of a generator $\alpha_G$ and its associated partition $\mathcal{A}$.

**Lemma 3.5.** *Assume that Assumption 3.1 is satisfied and $\mathscr{A}$ be a partition in $\mathbb{R}^d$. Then, any generator $G \in \mathcal{G}_L$ associated with $\mathcal{A}$ verifies:*

$$\alpha_G \leqslant 1 - \gamma(\partial^{\varepsilon_{min}} \mathcal{A}). \tag{3}$$

*where $\varepsilon_{min} = \min_{i,j} \|X_i - X_j\|/L$.*

Interestingly, this result holds independently of the partition $\mathcal{A}$. It highlights that the geometry of the partition gives an upper-bound on the precision of the generator. Consequently, to properly determine this bound on the precision levels of generative models, one might be interested in determining the measure of this epsilon-boundary $\partial^\varepsilon \mathcal{A}$. By using the result from Lemma 3.5, we can derive an upper-bound on the precision that depends on $D$, $L$ and $m$:

**Corollary 3.6.** *Assume that Assumption 3.1 is satisfied, $m \leqslant d + 1$. Then, there exists $L$ with $L \geqslant D\sqrt{\log(m)}$, such that for any well-balanced generator $G \in \mathscr{G}_L$:*

$$\alpha_G \leqslant 1 - \varepsilon_{min}\sqrt{\log m}\, e^{-3/2} \tag{4}$$

where $\varepsilon_{\min} = \min_{i,j} \|X_i - X_j\|/L$. In particular, the result in (4) gives an interesting insight when training GANs on a finite number of modes. Tanielian et al. (2020, Theorem 3) showed a similar result but for the asymptotic case when the number of modes increases:

$$\alpha_G \overset{m\to\infty}{\leqslant} e^{-\frac{1}{8}\varepsilon_{\min}^2} e^{-\varepsilon_{\min}\sqrt{\log(m)/2}}. \tag{5}$$

### 3.2. Optimality for push-forward generative models

To exhibit generative models with optimal precision levels, one must look at partitions with the smallest epsilon-boundary measures $\gamma(\partial^\varepsilon \mathcal{A})$. We argue that this is tightly

connected to the theoretical field of Gaussian isoperimetric inequalities. Isoperimetric inequalities link the measure of sets with their perimeters. More specifically, these inequalities highlight minimizers of the perimeter for a fixed measure, *e.g.* the sphere in an euclidean space with a given Lebesgue measure. In the Gaussian space, Borell (1975) and Sudakov and Tsirel'son (1978) show that in a finite-dimensional case, among all sets of a given measure, half-spaces have a minimum Gaussian perimeter. More formally, for any Borel set $A$ in $\mathbb{R}^d$ and a half-space $H$, if we have $\gamma(A) \geqslant \gamma(H)$, then $\gamma(A^\varepsilon) \geqslant \gamma(H^\varepsilon)$ for any $\varepsilon > 0$, where $A^\varepsilon$ denotes the $\varepsilon$-extension of $A$.

The Gaussian multi-bubble conjecture was formulated when looking for a way to partition the Gaussian space in $m$ parts, with the least-weighted boundary. It was recently proved by Milman and Neeman (2022) who showed that the best way to split a Gaussian space $\mathbb{R}^d$ in $m$ clusters of equal measure, with $2 \leqslant m \leqslant d+1$, is by using 'simplicial clusters' obtained as the Voronoi cells of $m$ equidistant points in $\mathbb{R}^d$. Convex geometry theory tells us that each cell is a convex cone, whose borders are hyperplanes going through the origin of $\mathbb{R}^d$. We note $\mathcal{A}^\star$ any partition corresponding to this optimal configuration, see Figure 1 for $m = 3$.

In the following theorem, we apply this result to the understanding of GANs. We make the connection between optimal generators (when $m \leqslant d+1$) in levels of precision and the partition $\mathcal{A}^\star$ derived in Milman and Neeman (2022).

**Theorem 3.7** (**Optimality of generators with simplicial cluster latent space.**). *Assume that Assumption 3.1 is satisfied and $m \leqslant d+1$. For any $\delta > 0$, there exists $C$ large enough (independent of $\delta$) and $L \geqslant D\sqrt{m}\sqrt{\pi \log(Cm)}$, and a well-balanced generator $G^\star \in \mathscr{G}_L$ associated with $\mathcal{A}^\star$ such that for any other well-balanced generator $G \in \mathscr{G}_L$, we have:*

$$\alpha_{G^\star} \geqslant \alpha_G - \delta \tag{6}$$

*Moreover, if $m \leqslant d$, noting $\varepsilon_{max} = \max_{i,j} \|X_i - X_j\|/L$:*

$$\alpha_{G^\star} \geqslant 1 - \varepsilon_{max}\sqrt{m \log(Cm)}, \tag{7}$$

Theorem 3.7 shows that when $L$ is large enough, the bound in (4) is almost tight, and thus that the given generator based on the simplicial partition $\mathcal{A}^\star$ is almost optimal. However, it is not clear whether those are the only generators with optimal precision. The proof is delayed in Appendix A.

**What if Assumption 3.1 is not verified?** This assumption is needed for the definition of a well-balanced generator associated with $\mathcal{A}^\star$ as in Theorem 3.7. As shown in Figure 2, the latent space configuration obtained by the GANs for 3 almost equidistant points (1st row) and 3 almost aligned data points (2nd row). We see that in the later case, the Voronoi partition of the target data points does not verify Assumption 3.1, and the optimal latent structure is not known. We observe in this specific case that it is made of two parallel hyperplanes, much different from $\mathcal{A}^\star$ defined by Milman and Neeman (2022) (1st row).

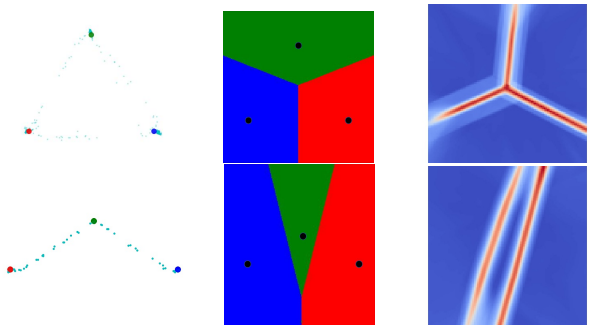

*Figure 2.* Illustration of the impact of the geometry of data modes on the latent space of GANs. The left column shows the modes $(X_1, X_2, X_3)$ from the target distribution and the generated points (small blue dots). In the middle, we plot the Voronoi diagram generated from $(X_1, X_2, X_3)$. On the right column, we show the boundaries in the GANs latent space with heatmaps of the norm of the gradient of the generator. In the first row, when the data satisfies Assumption 3.1, GANs achieve the optimal configuration. However, when the data modes do not satisfy this assumption, as seen in the second row, this is no longer the case.

**What if the dimension $m > d+1$?** The position of the different spheres could be such that Assumption 3.1 is no longer valid. Second, since the result from Milman and Neeman (2022) does not hold, the optimal partition of the Gaussian space in $m$ equal cells is unknown. In this generalized context, GANs could hint at the optimal partition geometry. Figure 3 stresses examples when training GANs from $\mathbb{R}^2$ to $\mathbb{R}^m$ with $m$ equidistant modes. This gives some insights on how to divide the Gaussian space into $m$ equitable areas with least Gaussian-weighted perimeter.

*Figure 3.* Extension of the multi-bubble conjecture when $m > d+1$. We depict the partition of the $\mathbb{R}^2$ latent space of a GAN that maps to $m$ equidistant points in $\mathbb{R}^m$, with $m = 4, 6, 12$. Each colored cell maps to a distinct data point in $\mathbb{R}^m$.

**What if the modes do not have equal measure?** The fact that each mode has equal measure in the target distribution might not be verified for unbalanced datasets. First, the optimality of simplicial clusters holds because the multi-bubble theorem is still valid. However, the

lower-bound (Equation 7) does not hold. Additionally, the upper-bound from Corollary 3.6 can be relaxed. Consider $w_1, \dots, w_m \in \mathbb{R}^m$ the weights of the different modes, and $w_{\min} = \min_i w_i$, the upper-bound becomes:

$$\alpha_G \leqslant 1 - m\varepsilon_{\min} w_{\min} \sqrt{\log(1/w_{\min})}\, e^{-3/2}.$$

We observe that this upper-bound might not be tight anymore since it depends on the minimum of the weights $w_{\min}$.

### 3.3. Improving generative models

Our proposed theoretical analysis offers valuable insights into the optimal structure of the latent space for push-forward generative models. We demonstrate that by leveraging this structure, it is possible to design GANs with improved performance. To achieve this, we enforce a simplicial cluster structure in the latent space of GANs during training using a novel rejection sampling procedure called *simplicial cluster truncation* that can be combined with a mutual-information loss. Note that modifying the latent space distribution of other generative models, such as VAEs or score-based models, is a more complex task.

**Simplicial cluster truncation.** Let us denote a simplicial cluster (Milman and Neeman, 2022) as $(u_1, \dots, u_m) \mid u_i \in \mathbb{R}^d$. The rejection sampling procedure, based on Theorem 3.7, involves sampling a latent vector $z$ from $\gamma$ and accepting it if $\max_{i \in [1,\dots,m]} (z \cdot u_i) > \tau$, where both $\tau$ and $m$ are considered as hyper-parameters. This defines a new latent space distribution where the density is high near the unit vectors $u_i, i \in [1, m]$. As a result, the boundaries of the simplicial cluster, which are points with high distances to the centers of Voronoi cells, are rejected. The threshold parameter $\tau$ determines the $\varepsilon$ value. With this method, the boundaries between different modes are never sampled, leading to a disconnected latent space. This approach can improve the learning of disconnected manifolds by injecting disconnectedness into the modeled generative distribution. Additionally, the use of a geometrical structure that is particularly well suited to separate several modes (Papyan et al., 2020) enhances the performance.

**Mutual-information loss.** The rejection sampling procedure might not be sufficient for the generator to properly use the different clusters of its latent space. To encourage the simplicial cluster structure, we also optimize the mutual information between generated samples and the corresponding cluster (Khayatkhoei et al., 2018). The loss is applied at the beginning of the training and is then dropped.

## 4. Experiments

In the following experiments[1], we validate our theoretical analysis and derive insights for GANs trained on toy and image datasets. We also run a small experiment on VAEs. We verify: 1) that the latent space geometry of GANs has similar properties than simplicial clusters; 2) that increasing the latent space dimension ($d + 1 > m$) can help improve GANs, as highlighted in the theoretical section; 3) that GANs' performance is correlated with their latent space geometry; 4) that the proposed *simplicial cluster* truncation method is effective and boost GANs' performance.

In the following experiments, we train WGANs with gradient penalty (Arjovsky et al., 2017; Gulrajani et al., 2017). For mixture of Gaussians, generator and discriminator are MLP networks. For MNIST, the generator and discriminator are standard convolutional architectures. On CIFAR-10, CIFAR-100 , and STL-10, we use either a Resnet-based (He et al., 2016) convolutional architecture with self-modulation in the generator (Chen et al., 2018b), either the transformer-based architecture from Jiang et al. (2021). To evaluate the performance of GANs, we use both the precision (Kynkäänniemi et al., 2019), the FID (Heusel et al., 2017), and the density/coverage (Naeem et al., 2020). We use a dataset-specific classifier to extract image features on MNIST, and InceptionNet pre-trained on ImageNet for CIFAR-10, CIFAR-100 and STL-10. Implementation details are given in Appendix B and code is provided in Supplementary Material.

### 4.1. Linear separability and convexity

According to Milman and Neeman (2022), the optimal configuration in the latent space is obtained as the Voronoi cells of $m$ equidistant points in $\mathbb{R}^d$, if $m \leq d+1$. This means that if GANs reach this optimal configuration, each of the cells must be *convex polytopes* and have the following properties: 1) the boundaries of a cell are flat; 2) each cell is convex. To investigate this, we use a labeled dataset and assess whether a simple linear model (*e.g.*, multinomial logistic regression) can map latents to labels. If the cells in the latent space are bounded by hyperplanes, then, using the hyperplane separation theorem, the linear model is expected to be a good predictor of a generated sample's label.

We use a standard multi-class labeled dataset. $G_\theta$ is a pre-trained generator and $C_\phi$ is a pre-trained classifier considered as an oracle. Using $G_\theta$ and $C_\phi$, we construct a dataset of latent vectors $z \in \mathbb{R}^d$ and their associated labels $y = C_\phi(G_\theta(z))$. On CIFAR-10/100, similarly to Razavi et al. (2019), only data points with a confidence threshold of 0.7 or higher are accepted. This dataset is later split into

[1]Our code is open-source and can be found there: https://github.com/thibautissenhuth/unveiling_latent_geometry.

| Dataset | Architecture | Latent dim | Precision (↑) | LogReg Acc. (↑) | Convex Acc. (↑) |
|---|---|---|---|---|---|
| 100 Gauss. | MLP | 100 | 75.5 | 78.5 | 87.2 |
| MNIST | CNN | 64 | 93.2 | 90.4 | 98.7 |
| CIFAR-10 | ResNet | 64 | 66.8 | 65.3 | 75.2 |
| CIFAR-10 | Transformer | 256 | 72.8 | 70.7 | 84.3 |
| CIFAR-100 | ResNet | 64 | 64.3 | 30.5 | 42.1 |
| CIFAR-100 | Transformer | 64 | 64.2 | 26.5 | 39.2 |

*Table 1.* Validation of linear separability (LogReg Acc.) and convexity (Convex Acc.) in GAN latent spaces. The results align with the predictions of Corollary 3.6, where a linearly separable and convex structure of the latent space indicates a high precision. The architecture *Transformer* refers to the TransGAN model from Jiang et al. (2021). The supervised classifiers used as oracles haves test-accuracies of 80.2% on CIFAR-10 and 61.8% on CIFAR-100.

| Dataset | LogReg Acc. (↑) | Convex Acc. (↑) |
|---|---|---|
| MNIST | 92.5 | 95.1 |
| CIFAR-10 | 53.1 | 59.4 |

*Table 2.* Validation of linear separability (LogReg Acc.) and convexity (Convex Acc.) in VAE latent spaces. The supervised classifiers used as oracles haves test-accuracies of 80.2% on CIFAR-10.

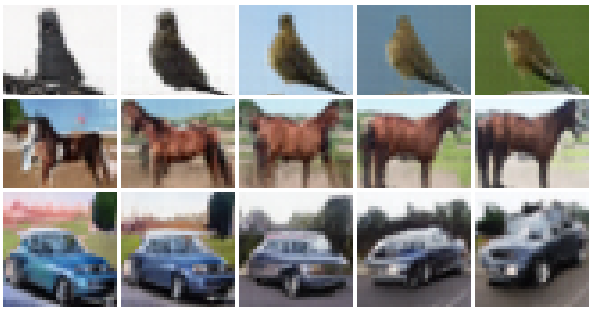

*Figure 4.* Visualization of the convexity of classes in the latent space of GANs trained on CIFAR-10. The plot shows that latent linear interpolations within a class preserve the class label.

100k training points and 10k test points. We use multinomial logistic regression to learn the mapping from latent vectors $z$ to their labels $y$. We can see in Table 1, that the *LogReg Accuracy* reaches high levels: 90% on MNIST and 70% on CIFAR-10. For the Convex accuracy, we draw two random latent vectors $z_0$ and $z_1$ that belong to the same class, and check whether linear interpolations in the latent space also belong to the same class, that is $C_\phi(G_\theta(z_0)) = C_\phi(G_\theta(z_0)) = \lambda \times C_\phi(G_\theta(z_0)) + (1-\lambda) \times C_\phi(G_\theta(z_1))$ for $\lambda \in [0,1]$. Interestingly, we see in Table 1 a correlation between the Logreg and Convex accuracy and the precision metric: the more the latent space behaves like a simplicial cluster, the higher the precision. For a qualitative evaluation, we show this phenomenon in Figure 4 and stress that linear interpolations conserve the image class.

In Table 2, we demonstrate that VAEs also tend to structure their latent space in linear regions. In this experiment, we use a slightly different setting: instead of drawing random latent vectors, we directly use the dataset samples and encode them with the VAE's encoder to test LogReg and Convex accuracies.

### 4.2. Impact of the latent space dimension

To evaluate the impact of the latent space dimension, we train GANs with latent space dimension ranging from 2 to 128 on several datasets. In Figure 5, we exhibit two phases in the performance of GANs when changing the number of latent dimensions. For a fixed architecture, and a given dataset, we observe the existence of an optimal latent space dimension $d^\star$. When $d < d^\star$ the precision or density of the model falls significantly. Interestingly, when $d > d^\star$, the precision becomes constant: overparameterizing the model does not bring a significant improvement. As expected, we observe in Figure 5 that the maximum precision/density depends on the complexity of the dataset and its number of modes: the more complex the dataset, the lower the precision. This is also coherent with our theoretical results from both Corollary 3.6 and Theorem 3.7.

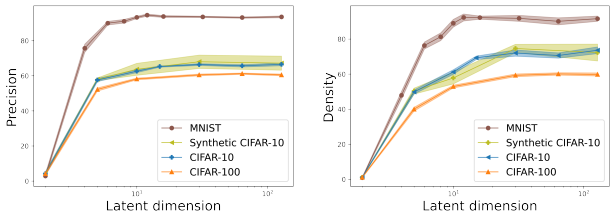


*Figure 5.* Performance of GANs with regard to the number of modes and latent space dimensions. As the number of modes and latent space dimension increases, we observe an improvement in Precision (left) and Density (right), with a saturation point beyond a certain threshold.

An interesting problem was also brought to the fore by Roth et al. (2017). When training GANs two different issues can arise: 1) *dimensional misspecification* where the true and modeled distributions do not have density functions

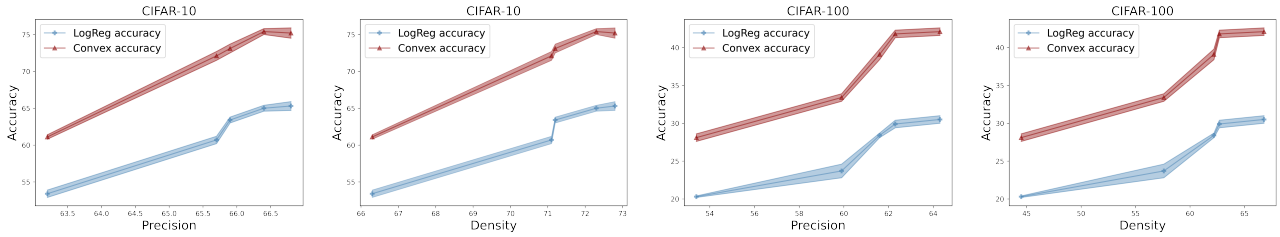


*Figure 6.* Study of the correlation between GANs' performance and their latent space geometry. This is done by increasing the width of the generator ($w \in \{32, 64, 128, 256, 512\}$) in a fixed training setting on the CIFAR-10 and CIFAR-100 datasets. The results reveal a positive correlation between GANs' performance (measured by Precision and Density) and the linear separability and convexity of their latent space (measured by LogReg and Convex Accuracy). Confidence intervals are computed on 10 checkpoints of a training.

w.r.t. the same base measure, and 2) *density misspecification*, where GANs try to fit a disconnected manifold with a unimodal distribution. To isolate the density misspecification studied in this paper, we train a conditional GAN with a low-dimensional latent space $\mathbb{R}^d$ (*e.g.* $\mathbb{R}^5$ in our setting), so that the dimension of the generated manifold is at most 5. We later collect a dataset of synthetic generated samples *Synthetic CIFAR-10*, and train unconditional GANs with varying latent space dimensions. Figure 5 shows that GANs converge to the same limits for Precision and Density on Synthetic CIFAR-10 and CIFAR-10. This shows that the performance is more impacted by the *density misspecification* (trying to fit a disconnected target distribution with a connected one) rather than the dimensional misspecification.

### 4.3. The latent geometry and GANs' performance

We investigate the relationship between the performance of GANs and their latent space geometry. To do so, we train many generators with different capacities (increasing width), and study how it impacts both the latent geometry and the performance. The results in Figure 6 reveal a strong positive correlation between the performance of GANs and the linearity/convexity of the latent space: the better the GANs perform, the more linearly separable and convex the latent space is. Indeed, the Pearson correlation between Precision and LogReg Accuracy is 0.98 on CIFAR-10, and 0.94 on CIFAR-100. Interestingly, overparametrization was known to help push-forward generative models in their optimization procedure (Balaji et al., 2020) and in increasing their Lipschitz constant (Salmona et al., 2022). We demonstrate here that it can help GANs in reaching an optimal latent space structure, resulting in improved performance.

### 4.4. Impact of the simplicial truncation method

Finally, we aim to improve GANs performance by using our theoretical results (Theorem 3.7). This is done by truncating the latent Gaussian distribution, as discussed in Section 3.3, so that the generator structures its latent space with a simplicial cluster geometry. Note that the rejection threshold used at inference time can be higher than the one used at training time, since we have observed that higher rejection thresholds can help us increase both the precision and density of the models. The results in Table 3 demonstrate that the use of this truncation method can improve the density and precision of GANs, without lowering the coverage nor the FID. This simplicial-based truncation has thus proved to be effective at removing off-manifold samples and can help improve push-forward generative models.

| Dataset/Model | FID ↓ | Prec. ↑ | Rec. ↑ | Dens. ↑ | Cov. ↑ |
|---|---|---|---|---|---|
| *CIFAR-10* | | | | | |
| TransGAN | 8.9 | 72.8 | **62.6** | 79.3 | 79.3 |
| TransGAN + JBT | **8.8** | 73.3 | 61.2 | 85.7 | 81.1 |
| TransGAN + DeliG. | 9.8 | 74.6 | 58.6 | 93.2 | 80.0 |
| **TransGAN + simp.** | 9.2 | **74.9** | 59.2 | **96.4** | **82.6** |
| *CIFAR-100* | | | | | |
| TransGAN | 15.2 | 64.2 | **63.1** | 53.4 | 66.0 |
| TransGAN + JBT | **15.0** | 64.8 | 62.9 | 53.6 | 66.2 |
| TransGAN + DeliG. | 15.9 | 63.5 | 62.2 | 52.6 | 64.4 |
| **TransGAN + simp.** | 15.1 | **65.6** | 61.5 | **56.3** | **66.4** |
| *STL-10 (32x32)* | | | | | |
| TransGAN | 10.5 | 75.7 | 60.1 | 87.5 | 83.0 |
| TransGAN + JBT | 11.0 | **78.1** | 57.6 | **99.3** | **83.8** |
| TransGAN + DeliG. | 10.5 | 76.0 | **60.2** | 85.5 | 81.5 |
| **TransGAN + simp.** | **10.0** | 77.8 | 60.1 | 94.1 | 83.5 |

*Table 3.* Improving GANs with simplicial cluster latent space. JBT stands for the Jacobian-based truncation (Tanielian et al., 2020); DeliG. for latent space with mixture of Gaussians (Gurumurthy et al., 2017); simp. for our proposed truncation method with simplicial cluster. These results demonstrate that generators with a simplicial cluster latent space consistently outperform the baseline generator in Precision/Density, and most of the times outperforms other boosting methods (DeliGAN and JBT).

## 5. Conclusion

In conclusion, this paper takes a step towards a better understanding of push-forward generative models. When the latent space dimension is large enough, we prove the exis-

tence of an optimal latent space geometry, called 'simplicial clusters'. Through experiments, we demonstrate that generative models with sufficient capacity tend to conform to this optimal geometry and also that enforcing this latent structure can improve GANs' performance. Our analysis has potential to drive further research on generative models with both theoretical and practical implications, such as developing new models that favor the emergence of such clusters in both latent and feature spaces. Similarly to what has been done in classification (Papyan et al., 2020), studying thorougly the feature space of deep generative models is also an open question.

**Limitations.** While our theoretical analysis demonstrates the existence of optimal generators, we were unable to prove their uniqueness. This limitation is associated with identifying partitions with the lowest $\varepsilon$-boundary measures in the Gaussian space, which is a challenging and unresolved problem in geometric measure theory.

**Potential negative societal impacts.** This work may increase potential negative impacts of deep generative models, such as *deepfakes* (Fallis, 2020).

## Acknowledgements

We would like to thank Jean-Yves Franceschi for helpful discussions and a thorough review of the paper.

## References


Rameen Abdal, Peihao Zhu, Niloy J Mitra, and Peter Wonka. Labels4free: Unsupervised segmentation using stylegan. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13970–13979, 2021.

Charu C Aggarwal, Alexander Hinneburg, and Daniel A Keim. On the surprising behavior of distance metrics in high dimensional space. In *International conference on database theory*, pages 420–434. Springer, 2001.

M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In D. Precup and Y.W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 214–223. PMLR, 2017.

Georgios Arvanitidis, Lars Kai Hansen, and Søren Hauberg. Latent space oddity: on the curvature of deep generative models. In *International Conference on Learning Representations*, 2018.

Samaneh Azadi, Catherine Olsson, Trevor Darrell, Ian Goodfellow, and Augustus Odena. Discriminator rejection sampling. In *International Conference on Learning Representations*, 2018.

Yogesh Balaji, Mohammadmahdi Sajedi, Neha Mukund Kalibhat, Mucong Ding, Dominik Stöger, Mahdi Soltanolkotabi, and Soheil Feizi. Understanding overparameterization in generative adversarial networks. In *International Conference on Learning Representations*, 2020.

Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? In *International conference on database theory*, pages 217–235. Springer, 1999.

Christer Borell. The brunn-minkowski inequality in gauss space. *Inventiones mathematicae*, 30(2):207–216, 1975.

A. Brock, J. Donahue, and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019.

Nutan Chen, Alexej Klushyn, Richard Kurle, Xueyan Jiang, Justin Bayer, and Patrick Smagt. Metrics for deep generative models. In *International Conference on Artificial Intelligence and Statistics*, pages 1540–1550. PMLR, 2018a.

Ting Chen, Mario Lucic, Neil Houlsby, and Sylvain Gelly. On self modulation for generative adversarial networks. In *International Conference on Learning Representations*, 2018b.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011.

Jeff Donahue and Karen Simonyan. Large scale adversarial representation learning. *Advances in neural information processing systems*, 32, 2019.

Don Fallis. The epistemic threat of deepfakes. *Philosophy & Technology*, pages 1–21, 2020.

Herbert Federer. *Geometric measure theory*. Springer, 1969.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and J. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A.C. Courville. Improved training of Wasserstein GANs. In I. Guyon, U. von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5767–5777. Curran Associates, Inc., 2017.

Swaminathan Gurumurthy, Ravi Kiran Sarvadevabhatla, and R Venkatesh Babu. Deligan: Generative adversarial networks for diverse and limited data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 166–174, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Steven Heilman, Aukosh Jagannath, and Assaf Naor. Solution of the propeller conjecture in $\mathbb{R}^3$. *Discrete & Computational Geometry*, 50(2):263–305, 2013.

M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

Ahmed Imtiaz Humayun, Randall Balestriero, and Richard Baraniuk. Polarity sampling: Quality and diversity control of pre-trained generative networks via singular values. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10641–10650, 2022.

Yifan Jiang, Shiyu Chang, and Zhangyang Wang. Transgan: Two pure transformers can make one strong gan, and that can scale up. *Advances in Neural Information Processing Systems*, 34, 2021.

T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems*, 34:852–863, 2021.

Mahyar Khayatkhoei, Maneesh K Singh, and Ahmed Elgammal. Disconnected manifold learning for generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 7343–7353, 2018.

Diederik P Kingma and Max Welling. Auto-encoding variational {Bayes}. In *International Conference on Learning Representations*, 2014.

N. Kodali, J. Abernethy, J. Hays, and Z. Kira. On convergence and stability of GANs. *arXiv.1705.07215*, 2017.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

T. Kynkäänniemi, T. Karras, S. Laine, J. Lehtinen, and T. Aila. Improved precision and recall metric for assessing generative models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 3927–3936. Curran Associates, Inc., 2019.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

Michel Ledoux. Isoperimetry and gaussian analysis. In *Lectures on probability theory and statistics*, pages 165–294. Springer, 1996.

Emanuel Milman and Joe Neeman. The gaussian double-bubble and multi-bubble conjectures. *Annals of Mathematics*, 195(1):89–206, 2022.

T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

Muhammad Ferjad Naeem, Seong Joon Oh, Youngjung Uh, Yunjey Choi, and Jaejun Yoo. Reliable fidelity and diversity metrics for generative models. *International Conference on Machine Learning*, pages 7176–7185, 2020.

Vardan Papyan, XY Han, and David L Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663, 2020.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019.

K. Roth, A. Lucchi, S. Nowozin, and T. Hofmann. Stabilizing training of generative adversarial networks through regularization. In I. Guyon, U. von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 2018–2028. Curran Associates, Inc., 2017.

M.S.M. Sajjadi, O. Bachem, M. Lucic, O. Bousquet, and S. Gelly. Assessing generative models via precision and recall. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 5228–5237. Curran Associates, Inc., 2018.

Antoine Salmona, Valentin De Bortoli, Julie Delon, and Agnès Desolneux. Can push-forward generative models fit multimodal distributions? In *Advances in Neural Information Processing Systems*, 2022.

Axel Sauer, Kashyap Chitta, Jens Müller, and Andreas Geiger. Projected gans converge faster. *Advances in Neural Information Processing Systems*, 34:17480–17492, 2021.

Gideon Schechtman. Approximate gaussian isoperimetry for k sets. In *Geometric Aspects of Functional Analysis*, pages 373–379. Springer, 2012.

Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9243–9252, 2020.

Vladimir N Sudakov and Boris S Tsirel'son. Extremal properties of half-spaces for spherically invariant measures. *Journal of Soviet Mathematics*, 9(1):9–18, 1978.

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

Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *Advances in Neural Information Processing Systems*, 33:19667–19679, 2020.

Aladin Virmaux and Kevin Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In *Advances in Neural Information Processing Systems*, pages 3835–3844, 2018.

X. Wei, B. Gong, Z. Liu, W. Lu, and L. Wang. Improving the improved training of Wasserstein GANs: A consistency term and its dual effect. In *International Conference on Learning Representations*, 2018.

Zongze Wu, Dani Lischinski, and Eli Shechtman. Stylespace analysis: Disentangled controls for stylegan image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12863–12872, 2021.

Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7354–7363, 2019.

Z. Zhou, J. Liang, Y. Song, L. Yu, H. Wang, W. Zhang, Y. Yu, and Z. Zhang. Lipschitz generative adversarial nets. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 7584–7593. PMLR, 2019.

Daniel Zoran, Rishabh Kabra, Alexander Lerchner, and Danilo J Rezende. Parts: Unsupervised segmentation with slots, attention and independence maximization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10439–10447, 2021.

# A. Technical results: proofs

## A.1. Proof of Lemma 3.5

We want to show that generator $G \in \mathcal{G}_L^{\mathcal{A}}$ is such that $\alpha_G \leqslant 1 - \gamma(\partial^{\varepsilon_{\min}}\mathcal{A})$, where

$$\partial^{\varepsilon_{\min}}\mathcal{A} = \bigcup_{i=1}^{m} \big(\cup_{j\neq i} A_j\big)^{\varepsilon_{\min}} \backslash \big(\cup_{j\neq i} A_j\big).$$

*Proof by contradiction.*

Assume a generator $G$ such that there exists $z \in \partial^{\varepsilon_{\min}}\mathcal{A}$ and $i \in [1,m]$ such that $G(z) \in M_i$. Since $G$ is associated with $\mathcal{A}$, we have using Definition 3.3, that there exists $z'$ and $j \in [1,m], j \neq i$ such that $\|z - z'\| < \varepsilon_{\min}/2$ and $j = \underset{k\in[1,m]}{\arg\min}\, \|G(z') - M_k\|$. Thus, we have:

$$\begin{aligned}
\|G(z) - G(z')\| &\geqslant d(G(z'), M_i),\\
&\geqslant d(M_i, M_i)/2,\\
&\geqslant D_{\min}/2.\\
\text{And, } \frac{\|G(z) - G(z')\|}{\|z - z'\|} &> D_{\min}/\varepsilon_{\min},\\
&> L.
\end{aligned}$$

This contradicts $G$ being in $\mathcal{G}_L^{\mathcal{A}}$.

## A.2. Proof of Corollary 3.6.

Let $L, D$ be such that $L \geqslant D\sqrt{\log(m)}$. Let's prove that for any well-balanced generator $G \in \mathcal{G}_L$, we have:

$$\alpha_G \leqslant 1 - \varepsilon_{\min}\sqrt{\log m}\, e^{-3/2}.$$

Using the method from Schechtman (2012), we have the measure of the border of cell $i$:

$$\begin{aligned}
\gamma\Big(\big(\cup_{j\neq i} A_j\big)^{\varepsilon} \backslash \big(\cup_{j\neq i} A_j\big)\Big) &\geqslant \frac{1}{\sqrt{2\pi}}\int_t^{t+\varepsilon} e^{-s^2/2} ds, \quad \text{where } t \text{ is such that } \frac{1}{\sqrt{2\pi}}\int_t^{\infty} e^{-s^2/2} ds = 1/m,\\
&\geqslant \frac{\varepsilon}{\sqrt{2\pi}} e^{-(t+\varepsilon)^2/2},\\
&\geqslant \frac{\varepsilon\sqrt{\log m}}{m} e^{-\varepsilon t - \varepsilon^2/2} \quad (\text{using} \sqrt{\log m} \leq t \leq \sqrt{2\log m}),\\
&\geqslant \frac{\varepsilon\sqrt{\log m}}{m} e^{-\varepsilon\sqrt{\log m} - \varepsilon^2/2}.
\end{aligned}$$

Thus:

$$\gamma(\partial^{\varepsilon_{\min}}\mathcal{A}) = \sum_{i=1}^{m} \gamma\Big(\big(\cup_{j\neq i} A_j\big)^{\varepsilon} \backslash \big(\cup_{j\neq i} A_j\big)\Big) \geqslant \varepsilon_{\min}\sqrt{\log m}\, e^{-\varepsilon_{\min}\sqrt{\log m} - \varepsilon_{\min}^2/2}.$$

Thus, we have

$$\begin{aligned}
\alpha_G &\leqslant 1 - \gamma(\partial^{\varepsilon_{\min}}\mathcal{A}),\\
&\leqslant 1 - \varepsilon_{\min}\sqrt{\log m}\, e^{-\varepsilon_{\min}\sqrt{\log m} - \varepsilon_{\min}^2/2}.
\end{aligned}$$

Moreover, using $\varepsilon_{\min} = \frac{D}{L}$ and $L \geqslant D\sqrt{\log m}$, so we get $\varepsilon_{\min}\sqrt{\log m} \leqslant 1$:

$$\alpha_G \leqslant 1 - \varepsilon_{\min}\sqrt{\log m}\, e^{-3/2}.$$

**Proof of Corollary 3.6 with $w_i$ (non-equal measure of modes).** Let $L, D$ be such that $L \geqslant D\sqrt{\log(m)}$. Let's prove that for any well-balanced generator $G \in \mathcal{G}_L$, we have:

$$\alpha_G \leqslant 1 - m w_{\min} \varepsilon_{\min} \sqrt{\log 1/w_{\min}}\, e^{-3/2}.$$

Using the method from Schechtman (2012), we have the measure of the border of cell $i$:

$$\begin{aligned}
\gamma\Big(\big(\cup_{j\neq i} A_j\big)^{\varepsilon} \backslash \big(\cup_{j\neq i} A_j\big)\Big) &\geqslant \frac{1}{\sqrt{2\pi}} \int_t^{t+\varepsilon} e^{-s^2/2} ds, \quad \text{where } t \text{ is such that } \frac{1}{\sqrt{2\pi}} \int_t^{\infty} e^{-s^2/2} ds = w_{\min}, \\
&\geqslant \frac{\varepsilon}{\sqrt{2\pi}} e^{-(t+\varepsilon)^2/2}, \\
&\geqslant w_{\min} \varepsilon \sqrt{\log 1/w_{\min}} e^{-\varepsilon t - \varepsilon^2/2} \quad (\text{using} \sqrt{\log 1/w_{\min}} \leq t \leq \sqrt{2 \log 1/w_{\min}}), \\
&\geqslant w_{\min} \varepsilon \sqrt{\log 1/w_{\min}} e^{-\varepsilon \sqrt{\log 1/w_{\min}} - \varepsilon^2/2}.
\end{aligned}$$

Thus:

$$\gamma(\partial^{\varepsilon_{\min}} \mathcal{A}) = \sum_{i=1}^{m} \gamma\Big(\big(\cup_{j\neq i} A_j\big)^{\varepsilon} \backslash \big(\cup_{j\neq i} A_j\big)\Big) \geqslant m w_{\min} \varepsilon_{\min} \sqrt{\log 1/w_{\min}}\, e^{-\varepsilon_{\min}\sqrt{\log 1/w_{\min}} - \varepsilon_{\min}^2/2}.$$

Thus, we have

$$\begin{aligned}
\alpha_G &\leqslant 1 - \gamma(\partial^{\varepsilon_{\min}} \mathcal{A}), \\
&\leqslant 1 - m w_{\min} \varepsilon_{\min} \sqrt{\log 1/w_{\min}}\, e^{-\varepsilon_{\min}\sqrt{\log 1/w_{\min}} - \varepsilon_{\min}^2/2}.
\end{aligned}$$

Moreover, using $\varepsilon_{\min} = \frac{D}{L}$ and $L \geqslant D\sqrt{\log 1/w_{\min}}$, so we get $\varepsilon_{\min}\sqrt{\log 1/w_{\min}} \leqslant 1$:

$$\alpha_G \leqslant 1 - m \varepsilon_{\min} w_{\min} \sqrt{\log 1/w_{\min}}\, e^{-3/2}.$$

### A.3. Proof of Theorem 3.7

Let $\mu^\star$ be the target distribution. We know that $\mu_\star$ lays on m disconnected components contained in spheres $S_i, i \in [1, m]$. We note $M_i, i \in [1, m]$ the centers, and $r_i$ the radius of each sphere. We also assume that the spheres verify Assumption 3.1. For each pair $(i, j) \in [1, m]^2$, we define $X_{ij} \in S_i$ and $X_{ji} \in S_j$ the points verifying

$$X_{ij} = \underset{x \in S_i}{\arg\min}\ d(x, S_j) \quad \text{and} \quad X_{ji} = \underset{x \in S_j}{\arg\min}\ d(x, S_i).$$

We consider the optimal partition $\mathcal{A}^\star$ in the Gaussian latent space. For each given latent point $z \in \mathbb{R}^d$, we define:

$$N_z = \{j \in [1, m], z \in A_j^{\varepsilon}\}.$$

We then distinguish two different cases:

1. $|N_z| = 1$: the point $z$ belongs to the interior of a single cell, $z \in A_i^{-\varepsilon}$.
2. $|N_z| \geqslant 2$: the point $z$ is in the neighborhood of at least two different cells.

Interestingly, a point can only belong at most to the interior of one cell, but it can be at the intersection of several boundaries. We are now ready to define the optimal generator.

First, we set

$$G(z) = X_{i,j} \text{ for all } z \in \{z \in \mathbb{R}^d, |N_z| = 2, z \in \overline{A_i^{-\varepsilon}} \cap A_i^{\varepsilon} \cap A_j^{\varepsilon} \text{ where } N_z = \{i, j\}\}.$$

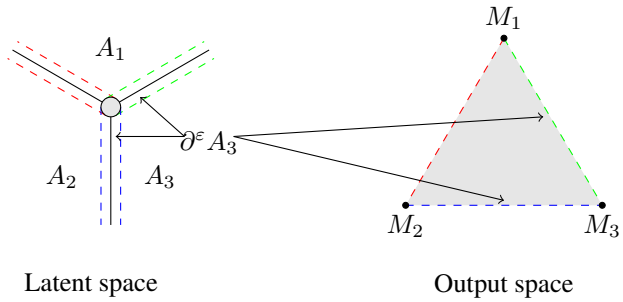


*Figure 7.* An optimal generator maps a 2D latent space to a 2D output space with three modes $(M_1, M_2, M_3)$. The latent space has an optimal 'simplicial cluster' geometry. In the latent space, all the $\epsilon$-boundaries intersect each other in the gray circle, which is mapped in the output space in the convex hull of the three modes.

Second, we define the generator in the interior of the cells, i.e. $N_z = \{i\}$. For each $z \in A_i^{-\varepsilon}$ and for a given unit vector $u \in \mathbb{R}^d$, we assume that the generator is constant along the parametric line $z = k \times u, k \in \mathbb{R}$.

Finally, we define the generator when $z$ does not belong to the interior of any cell, i.e. $|N_z| \geqslant 2$:

$$G^\star_\varepsilon(z) = \sum_{i\in[1,m]} \sum_{j\neq i} w_{i,j}(z)\, X_{i,j}\, \mathbb{1}_{j\in N_z}\, \mathbb{1}_{i\in N_z} \quad \text{where} \quad w_{i,j}(z) = \frac{d(z,(A_i^\varepsilon)^\complement)}{\sum_{i\in[1,m]}\sum_{j\neq i} d(z,(A_j^\varepsilon)^\complement)\, \mathbb{1}_{j\in N_z}\, \mathbb{1}_{i\in N_z}} \tag{8}$$

where $d(z, A) = \min_{a\in A} \|z - a\|$. An illustration of the optimal generator is given in Figure 7. When z belongs to the intersection of two $\varepsilon$-boundaries, $G_\varepsilon(z)$ is a simple linear combination of 2 points. It is only when $|N_z| \geqslant 3$ that more complex samples are generated. A simple illustration of $G^\star_\varepsilon$ for $d = 2$ and $m = 3$ is given in Figure 7. Interestingly, one can also show that the image of $G^\star_\varepsilon$ is equal to the convex hull of the Diracs $X_i, i \in [1, m]$. In particular, there exists a particularly interesting neighborhood $\nu$ of 0 where $G^\star_\varepsilon(\nu)$ is equal to the whole convex hull of the points $X_i, i \in [1, m]$.

**Proof that $G^\star_\varepsilon$ is well-balanced.** We recall that a generator is *well-balanced* if we have $G\sharp\gamma(M_1) = \ldots = G\sharp\gamma(M_m)$. By construction (8), we have that for any $i \in [1, m]$

$$\begin{aligned}\|G^\star_\varepsilon(z) - X_i\| &= \|\sum_{k\neq i} w_k(X_k - X_i)\|,\\ &= D \times (1 - w_i).\end{aligned}$$

So, for any $z \in A_i$, we have that

$$i = \underset{j\in[1,m]}{\arg\min}\ w_j = \underset{j\in[1,m]}{\arg\min}\ \|G(z) - X_j\|$$

Thus $G^\star_\varepsilon$ is associated with the optimal partition $\mathcal{A}^\star$, .

Besides, for a given radius $r$ of the different modes, since everything is symmetrical, we have that $\gamma(\{z \in \mathbb{R}^d, \|G(z) - X_1\| \leqslant r\} = \ldots = \gamma(\{z \in \mathbb{R}^d, \|G(z) - X_m\| \leqslant r\}$. Thus, the generator is well-balanced.

**Showing that $G^\star_{\varepsilon^\star}$ is in $\mathcal{G}_L$.** It is clear that when $|N_z| = 1$, we have that $G^\star_\varepsilon(z)$ is a $L$-Lipshitz continuous function.

Now, assume that $|N_z| \geqslant 2$. Consider $z, z'$ such that $N_z = N'_z$. Let $\alpha = (\alpha_1, \ldots, \alpha_m)$ and $\beta = (\beta_1, \ldots, \beta_m)$ be two vectors, both in $\mathbb{R}^m$, such that for all $i \in [1, m]$:

$$\alpha_i = \frac{d(z,(A_i^\varepsilon)^\complement)}{\sum_{j\in\mathcal{A}_z} d(z,(A_j^\varepsilon)^\complement)} \quad \text{and} \quad \beta_i = \frac{d(z',(A_i^\varepsilon)^\complement)}{\sum_{j\in\mathcal{A}_z} d(z',(A_j^\varepsilon)^\complement)}$$

We have that

$$\begin{aligned}\|G(z)-G(z')\| &= \|(1-\sum_{i\neq 1}\alpha_i)X_1-(1-\sum_{i\neq 1}\beta_i)X_1+\sum_{i\neq 1}\alpha_i X_i-\sum_{i\neq 1}\beta_i X_i\| \\ &= \|\sum_{i\neq 1}(\alpha_i-\beta_i)(X_1-X_i)\| \\ &\leqslant \max_{(i,j)\in[1,m]^2}\|X_i-X_j\|\,\|\alpha-\beta\|, \\ &\leqslant \max_{(i,j)\in[1,m]^2}\|X_i-X_j\|\,\|h(z)-h(z')\|,\end{aligned}$$

where $h$ is the function from $\mathbb{R}^d \to \mathbb{R}^m$ defined as:

$$h(z) = \left(\frac{d(z,(A_1^\varepsilon)^{\complement})}{\sum_{i\in\mathcal{A}_z} d(z,(A_i^\varepsilon)^{\complement})},\dots,\frac{d(z,(A_m^\varepsilon)^{\complement})}{\sum_{i\in\mathcal{A}_z} d(z,(A_i^\varepsilon)^{\complement})}\right).$$

We can write $h = f\circ g$ with $f$ the function defined from $\mathbb{R}^d \to \mathbb{R}^m$ by

$$f(z) = \Big(d(z,(A_1^\varepsilon)^{\complement}),\dots,d(z,(A_k^\varepsilon)^{\complement})\Big),$$

and $g$ the function defined on $\mathbb{R}^m \setminus \{0\}$ by

$$g(z) = \frac{z}{\|z\|_1}$$

We have that $f$ is a $\sqrt{m}$-Lipschitz functions (given that $z \mapsto d(z,(A_m^\varepsilon)^{\complement})$ is 1-Lipschitz). Besides, we know that outside the ball $B_{\varepsilon/2}(0)$, the function $g$ is $(2/\varepsilon)$-Lipschitz. Since it is clear that for every point $z$ such that $|N_z| \geqslant 2$, we have that $|f(z)| \geqslant \varepsilon/2$. Finally, the function $h$ is $\frac{2\sqrt{m}}{\varepsilon}$-Lipschitz. Thus, we have that:

$$\|G_\varepsilon^\star(z)-G_\varepsilon^\star(z')\| \leqslant \frac{2D\sqrt{m}}{\varepsilon}\|z-z'\|,$$

with $D = \max_{i,j}\|X_i-X_j\|, (i,j)\in[1,m]^2, i\neq j$.

Now, by noting $\varepsilon_{\max} = \frac{D}{L}$, and considering $\varepsilon^\star = 2\sqrt{m}\,\varepsilon_{\max}$, we have:

$$\|G_{\varepsilon^\star}^\star(z)-G_{\varepsilon^\star}^\star(z')\| \leqslant L\|z-z'\|.$$

Now, consider two latent vectors $z, z'$ in the same cell $\overline{A_i^{-\varepsilon}}$. There exists $i\in[1,m]$, and a pairs $(j,j')\in[1,m]^2$ (note that $j$ could be equal to $j'$) such that $G(z) = X_{i,j}$ and $G(z') = X_{i,j'}$. Using a similar reasoning as before, we can show that:

$$\|G_{\varepsilon^\star}^\star(z)-G_{\varepsilon^\star}^\star(z')\| \leqslant L\|z-z'\|,$$

with $D = 2\max_{i\in[1,m]}\ r_i$.

We can now conclude on the Lipschitzness of $G^\star$ on $\mathbb{R}^d$.

**Proving that: for $m \leqslant d+1$, for any $\delta > 0$, if $L$ is large enough, then, for any well-balanced $G \in \mathcal{G}_L$, we have $\alpha_{G^\star_{\varepsilon_{\max}}} \geqslant \alpha_G - \delta$.** Let $G$ be a well-balanced generator and $\mathcal{A}$ the partition associated with $G$. Let us first define the gaussian boundary measure $P_\gamma$ of a partition $\mathcal{A}$ of $\mathbb{R}^d$. For partitions with smooth boundaries, it coincides with the $(d-1)$-dimensional gaussian measure of the boundary, defined as follows:

$$P_\gamma(\mathcal{A}) = \liminf_{\varepsilon\to 0}\frac{\gamma(\partial^\varepsilon\mathcal{A})-\gamma(\mathcal{A})}{\sqrt{2/\pi}\varepsilon}$$

Moreover, for sets with smooth boundaries, we have from Federer (1969, Theorem 3.2.29):

$$\liminf_{\varepsilon\to 0}\frac{\gamma(\partial^\varepsilon\mathcal{A})-\gamma(\mathcal{A})}{\sqrt{2/\pi}\varepsilon} = \lim_{\varepsilon\to 0}\frac{\gamma(\partial^\varepsilon\mathcal{A})-\gamma(\mathcal{A})}{\sqrt{2/\pi}\varepsilon}$$

Let us denote $\mathcal{A}^\star$, the optimal partition defined in Milman and Neeman (2022), based on simplicial clusters. $A^\star$ is a standard partition where $\gamma(A_1^\star) = \ldots = \gamma(A_m^\star)$ for all i, and $\sum_i \gamma(A_i) = 1$. By the multi-bubble theorem (Milman and Neeman, 2022), simplicial clusters (such as $\mathcal{A}^\star$) are the unique minimizers of the gaussian isoperimetric problem, thus:

$$\begin{aligned} P_\gamma(\mathcal{A}^\star) &\leqslant P_\gamma(\mathcal{A}) \\ \lim_{\varepsilon\to 0} \frac{\gamma(\partial^\varepsilon \mathcal{A}^\star)}{\varepsilon} &\leqslant \lim_{\varepsilon\to 0} \frac{\gamma(\partial^\varepsilon \mathcal{A})}{\varepsilon} \\ L_{\mathcal{A}} &\leqslant L_{\mathcal{A}^\star} \end{aligned}$$

where $L_{\mathcal{A}} = \lim_{\varepsilon\to 0} \frac{\gamma(\partial^\varepsilon \mathcal{A}^\star)}{\varepsilon}$ and $L_{\mathcal{A}^\star} = \lim_{\varepsilon\to 0} \frac{\gamma(\partial^\varepsilon \mathcal{A}^\star)}{\varepsilon}$.

Then, for any $\delta > 0$, there exists $\varepsilon' > 0$ such that, for any $\varepsilon < \varepsilon'$,

$$|\frac{\gamma(\partial^\varepsilon \mathcal{A}^\star)}{\varepsilon} - L_{\mathcal{A}^\star}| < \delta \quad , \quad |\frac{\gamma(\partial^\varepsilon \mathcal{A})}{\varepsilon} - L_{\mathcal{A}}| < \delta \quad \text{and} \quad L_{\mathcal{A}^\star} \leqslant L_{\mathcal{A}}$$

Thus, for any $\delta > 0$, there exists $\varepsilon' > 0$ such that, for any $\varepsilon < \varepsilon'$,

$$\gamma(\partial^\varepsilon \mathcal{A}^\star) \leqslant \gamma(\partial^\varepsilon \mathcal{A}) + 2\delta\varepsilon \tag{9}$$

Besides, we know that

$$\alpha_G \leqslant 1 - \gamma(\partial^{\varepsilon_{\text{min}}} \mathcal{A})$$

Consequently, we have that:

$$\begin{aligned} \alpha_G &\leqslant 1 - \gamma(\partial^{\varepsilon_{\text{min}}} \mathcal{A}) \\ &\leqslant 1 - \gamma(\partial^{\varepsilon_{\text{min}}} \mathcal{A}^\star) + 2\delta\varepsilon_{\text{min}} \quad \text{using (9).} \end{aligned}$$

Now, by construction of $G^\star_{\varepsilon_{\text{max}}}$, we have that

$$\alpha_{G^\star_{\varepsilon_{\text{max}}}} \geqslant 1 - \gamma(\partial^{\varepsilon_{\text{max}}} \mathcal{A}^\star).$$

Consequently,

$$\begin{aligned} \alpha_G &\leqslant 1 - \gamma(\partial^{\varepsilon_{\text{min}}} \mathcal{A}^\star) + 2\delta\varepsilon_{\text{max}} + \gamma(\partial^{\varepsilon_{\text{max}}} \mathcal{A}^\star) - \gamma(\partial^{\varepsilon_{\text{max}}} \mathcal{A}^\star) \\ &\leqslant \alpha_{G^\star_\varepsilon} + 2\delta\varepsilon_{\text{max}} + \gamma(\partial^{\varepsilon_{\text{max}}} \mathcal{A}^\star) - \gamma(\partial^{\varepsilon_{\text{min}}} \mathcal{A}^\star) \\ &\leqslant \alpha_{G^\star_\varepsilon} + 2\delta\varepsilon_{\text{max}} + \gamma(\partial^{\varepsilon_{\text{max}}} \mathcal{A}^\star) - 2L_{\mathcal{A}^\star}\varepsilon_{\text{max}} - \gamma(\partial^{\varepsilon_{\text{min}}} \mathcal{A}^\star) + 2L_{\mathcal{A}^\star}\varepsilon_{\text{min}} + 2L_{\mathcal{A}^\star}(\varepsilon_{\text{max}} - \varepsilon_{\text{min}}) \\ &\leqslant \alpha_{G^\star_\varepsilon} + 4\delta\varepsilon_{\text{max}} + 2L_{\mathcal{A}^\star}\varepsilon_{\text{max}}, \\ &\leqslant \alpha_{G^\star_\varepsilon} + \varepsilon_{\text{max}}(4\delta + 2L_{\mathcal{A}^\star}). \end{aligned}$$

We conclude by choosing $L$ big enough such that $\varepsilon_{\text{max}}$ is strictly smaller than $\frac{\delta}{4\delta + 2L_{\mathcal{A}^\star}}$.

**Proving the lower-bound 7 of Theorem 3.7.** Let's consider $G_{\varepsilon^\star}$ defined using (8) and $\varepsilon^\star = 2\sqrt{m}\varepsilon_{\text{max}}$. The precision of $G^\star_{\varepsilon^\star}$ is thus such that:

$$\alpha_{G^\star_{\varepsilon^\star}} \geqslant 1 - \gamma(\partial^{\varepsilon^\star} \mathcal{A}).$$

However, since $\partial^\varepsilon \mathcal{A} \subset \bigcup_{i=1}^n A_i^\varepsilon$, we have that for any $\varepsilon$:

$$\gamma(\partial^\varepsilon \mathcal{A}) \leqslant \sum_{i=1}^n \gamma(A_i^\varepsilon).$$

Using results from Schechtman (2012, Proposition 1), when $m \leq d$, there exists $C$ large enough, such that

$$\gamma(A_i^{\varepsilon^\star}) \leqslant \frac{\varepsilon^\star}{m}\big(\sqrt{\pi \log(Cm)}\big).$$

Thus, we have

$$\alpha_{G^\star_{\varepsilon^\star}} \geqslant 1 - \varepsilon^\star \sqrt{\pi \log(Cm)},$$

To have $\alpha_{G^\star_{\varepsilon_{\max}}} \geqslant 0$, we must have $^\star \leqslant 1/\sqrt{\pi \log(Cm)}$. This is the case since we have

$$^\star = 2D\sqrt{m}/L \quad \text{and} \quad L \geqslant D\sqrt{m}\sqrt{\pi \log(Cm)},$$

where $D = \max_{i,j} \|X_i - X_j\|$.

# B. Experiments

## B.1. Implementation details

*Table 4.* GANs training details on MNIST

| Operation | Kernel | Strides | Feature Maps | Activation |
|---|---|---|---|---|
| Generator G(z) | | | | |
| $z \sim \mathrm{N}(0, I)$ | | | $dim(z)$ | |
| Fully Connected | | | $7 \times 7 \times 128$ | |
| Convolution | $3 \times 3$ | $1 \times 1$ | $7 \times 7 \times 64$ | LReLU |
| Convolution | $3 \times 3$ | $1 \times 1$ | $7 \times 7 \times 64$ | LReLU |
| Nearest Up Sample | | | $14 \times 14 \times 64$ | |
| Convolution | $3 \times 3$ | $1 \times 1$ | $14 \times 14 \times 32$ | LReLU |
| Convolution | $3 \times 3$ | $1 \times 1$ | $14 \times 14 \times 32$ | LReLU |
| Nearest Up Sample | | | $14 \times 14 \times 32$ | |
| Convolution | $3 \times 3$ | $1 \times 1$ | $28 \times 28 \times 16$ | LReLU |
| Convolution | $3 \times 3$ | $1 \times 1$ | $28 \times 28 \times 1$ | Tanh |
| D(x) | | | $28 \times 28 \times 1$ | |
| Convolution | $4 \times 4$ | $2 \times 2$ | $14 \times 14 \times 512$ | LReLU |
| Convolution | $3 \times 3$ | $1 \times 1$ | $14 \times 14 \times 512$ | LReLU |
| Convolution | $4 \times 4$ | $2 \times 2$ | $7 \times 7 \times 512$ | LReLU |
| Convolution | $3 \times 3$ | $1 \times 1$ | $7 \times 7 \times 512$ | LReLU |
| Fully Connected | | | 1 | - |
| Batch size | 256 | | | |
| Leaky ReLU slope | 0.2 | | | |
| Gradient Penalty weight | 10 | | | |
| Learning Rate Discriminator | $1 \times 10^{-4}$ | | | |
| Learning Rate Generator | $5 \times 10^{-5}$ | | | |
| Disciminator steps | 2 | | | |
| Optimizer | Adam | $\beta_1 : 0.5$ | $\beta_2 : 0.5$ | |

First, let us note that we share our code in Supplementary Material for reproducibility.

**Training.** We use the Wasserstein loss with gradient-penalty on interpolations of fake and real data. At each iteration, the discriminator is trained 2 steps and the generator 1 step with Adam optimizer. The batch size is 256. The learning rate of the discriminator is two times larger (Heusel et al., 2017), *i.e.* $5 \times 10^{-5}$ for the generator and $1 \times 10^{-4}$ for the discriminator. GANs are trained for 80k steps on MNIST and for 100k steps on CIFAR datasets. Architectures of generator and discriminator are described in Table 4 and Table 5.

For TransGAN (Jiang et al., 2021), we follow the implementation from the authors available at https://github.com/VITA-Group/TransGAN. TransGAN is trained with a WGAN-GP loss, 4 discriminator steps for 1 generator step, and Adam optimizer with a learning rate of $10^{-4}$.

**Evaluation.** For evaluation metrics, we follow the setting proposed by the authors. For FID (Heusel et al., 2017), we use 50k real images and 50k fake images. For precision, recall, density and coverage (Kynkäänniemi et al., 2019; Naeem et al., 2020), we use 10k real images and 10k fake images with nearest-k$= 5$.

*Table 5.* GANs training details on CIFAR datasets. BN stands for batch-normalization.

| Operation | Kernel | Strides | Feature Maps | Conditional BN (Chen et al., 2018b) | Activation |
|---|---|---|---|---|---|
| Generator G(z) | | | | | |
| $z \sim \mathrm{N}(0, Id)$ | | | 128 | | |
| Fully Connected | | | $4 \times 4 \times 128$ | - | |
| ResBlock | $[3 \times 3] \times 2$ | $1 \times 1$ | $4 \times 4 \times 128$ | Y | ReLU |
| Nearest Up Sample | | | $8 \times 8 \times 128$ | - | |
| ResBlock | $[3 \times 3] \times 2$ | $1 \times 1$ | $8 \times 8 \times 128$ | Y | ReLU |
| Nearest Up Sample | | | $16 \times 16 \times 128$ | - | |
| ResBlock | $[3 \times 3] \times 2$ | $1 \times 1$ | $16 \times 16 \times 128$ | Y | ReLU |
| Nearest Up Sample | | | $32 \times 32 \times 128$ | - | |
| Convolution | $3 \times 3$ | $1 \times 1$ | $32 \times 32 \times 3$ | - | Tanh |
| Discriminator D(x) | | | $32 \times 32 \times 3$ | | |
| ResBlock | $[3 \times 3] \times 2$ | $1 \times 1$ | $32 \times 32 \times 256$ | - | ReLU |
| AvgPool | $2 \times 2$ | $1 \times 1$ | $16 \times 16 \times 256$ | - | |
| ResBlock | $[3 \times 3] \times 2$ | $1 \times 1$ | $16 \times 16 \times 256$ | - | ReLU |
| AvgPool | $2 \times 2$ | $1 \times 1$ | $8 \times 8 \times 256$ | - | |
| ResBlock | $[3 \times 3] \times 2$ | $1 \times 1$ | $8 \times 8 \times 256$ | - | ReLU |
| ResBlock | $[3 \times 3] \times 2$ | $1 \times 1$ | $8 \times 8 \times 256$ | - | ReLU |
| Mean spatial pooling | - | - | 256 | - | |
| Fully Connected | | | 1 | - | - |
| Batch size | 256 | | | | |
| Gradient Penalty weight | 10 | | | | |
| Learning Rate Discriminator | $1 \times 10^{-4}$ | | | | |
| Learning Rate Generator | $5 \times 10^{-5}$ | | | | |
| Discriminator steps | 2 | | | | |
| Optimizer | Adam | $\beta_1 = 0.$ | $\beta_2 = 0.999$ | | |

**GPUs.** For all datasets, the training of GANs was run on NVIDIA Tesla V100 GPUs (16 GB). The training of ResNet GANs for 100k steps on CIFAR takes around 30 hours. For TransGAN models, the training is done for 250k steps on two NVIDIA Tesla V100 GPUs, which takes around $35 \times 2 = 70$ GPU hours.

### B.2. Correlation between latent space geometry and GANs' performance (Details for Section 4.3)

We present the full results of this study in Table 6.

| Dataset | Width | LogReg Acc. ↑ | Convex Acc. ↑ | FID ↓ | Prec. ↑ | Rec. ↑ | Dens. ↑ | Cov. ↑ |
|---|---|---|---|---|---|---|---|---|
| CIFAR-10 (Resnet) | 32 | $53.4 \pm 0.5$ | $61.1 \pm 0.3$ | $28.3 \pm 0.6$ | $63.2 \pm 0.6$ | $58.6 \pm 0.9$ | $66.3 \pm 1.5$ | $61.3 \pm 1.1$ |
| | 64 | $60.7 \pm 0.5$ | $72.1 \pm 0.6$ | $20.6 \pm 0.3$ | $65.7 \pm 0.5$ | $62.0 \pm 0.6$ | $71.4 \pm 1.7$ | $71.5 \pm 1.0$ |
| | 128 | $63.4 \pm 0.4$ | $73.1 \pm 0.6$ | $17.0 \pm 0.3$ | $65.9 \pm 0.4$ | $64.5 \pm 0.9$ | $71.2 \pm 1.5$ | $74.6 \pm 0.9$ |
| | 256 | $65.0 \pm 0.4$ | $\mathbf{75.4} \pm 0.4$ | $\mathbf{16.1} \pm 0.3$ | $66.4 \pm 0.5$ | $\mathbf{66.2} \pm 1.0$ | $72.3 \pm 1.5$ | $75.6 \pm 1.0$ |
| | 512 | $\mathbf{65.3} \pm 0.6$ | $75.2 \pm 0.7$ | $\mathbf{16.1} \pm 0.3$ | $\mathbf{66.8} \pm 1.0$ | $66.1 \pm 1.3$ | $\mathbf{72.8} \pm 2.9$ | $\mathbf{76.1} \pm 1.4$ |
| CIFAR-100 (Resnet) | 32 | $20.3 \pm 0.1$ | $28.1 \pm 0.5$ | $28.3 \pm 0.3$ | $53.4 \pm 0.7$ | $63.5 \pm 0.8$ | $44.5 \pm 1.3$ | $56.1 \pm 1.2$ |
| | 64 | $23.7 \pm 0.9$ | $33.4 \pm 0.5$ | $23.4 \pm 0.3$ | $59.9 \pm 0.5$ | $64.6 \pm 0.7$ | $57.6 \pm 1.7$ | $67.6 \pm 0.5$ |
| | 128 | $28.4 \pm 0.3$ | $39.1 \pm 0.7$ | $21.1 \pm 0.4$ | $61.6 \pm 0.4$ | $63.8 \pm 0.5$ | $62.2 \pm 1.0$ | $70.2 \pm 0.5$ |
| | 256 | $29.9 \pm 0.5$ | $41.8 \pm 0.6$ | $21.0 \pm 0.4$ | $62.3 \pm 0.6$ | $\mathbf{65.6} \pm 0.6$ | $62.7 \pm 2.0$ | $70.1 \pm 0.9$ |
| | 512 | $\mathbf{30.5} \pm 0.5$ | $\mathbf{42.1} \pm 0.5$ | $\mathbf{19.7} \pm 0.4$ | $\mathbf{64.3} \pm 0.8$ | $64.8 \pm 0.8$ | $\mathbf{66.8} \pm 1.9$ | $\mathbf{72.2} \pm 0.6$ |

*Table 6.* Correlation between GANs' performance and their latent space geometry. Increasing the capacity of GANs tend to structure their latent space in simplicial clusters (better LogReg accuracy) and improve their performance on precision, density and coverage. Confidence intervals are computed on several sets of generated/training points from a given generator.

### B.3. Details on simplicial truncation method (Details for Section 4.4)

We provide here more details about our truncation method. First, the rejection sampling in the latent space $\mathbb{R}^d$ of GANs procedure is the following:

- Define hyper-parameters threshold $\tau$, number of clusters $N$, latent space dimension $d$.
- Initialize N equidistant points in $\{(u_0, \dots, u_N) \mid u_i \in \mathbb{R}^d\}$. This can be done easily when $N \leq d$.
- When sampling latent vectors $z \in \mathbb{R}^d$, compute a softmax over the negative distances between $z$ and $u_i$: $p_i(z) = \frac{e^{-d(z,u_i)}}{\sum_j e^{-d(z,u_j)}}$. Then, z is selected if $\max_i \big(p_i(z)\big) > \tau$.

Second, we add a classification loss to encourage the generator to use this latent structure. This loss is motivated by the need to maximize mutual information between the latent cluster and the modes of the generator (Khayatkhoei et al., 2018), and can be written as:

$$L_c = -\mathbb{E}_{z\sim\gamma}[\ln q_\phi\big(i(z)|G_\theta(z)\big)]$$

where $q_\phi$ is parametrized by a second classification head added to the discriminator; $i(z) = \arg\max_i \big(p_i(z)\big)$ is the index of the latent cluster of the sample. This loss is added during training, at each step of generator's and discriminator's training, during the first 20 epochs. It is then dropped, since we noticed that it harms the GANs performance if it is kept until the end of the training.

Training hyper-parameters: for $N = 10$, we use a latent dimension of $d = 64$ and training threshold of $\tau = 0.135$; for $N = 100$, we use $d = 128$ and $\tau = 0.08$.

During inference, if the generator has properly learned to use the different clusters of the latent space, we observe that augmenting the threshold $\tau$ leads to an increased density and precision.

We present full results in Table 7 and Figure 8.

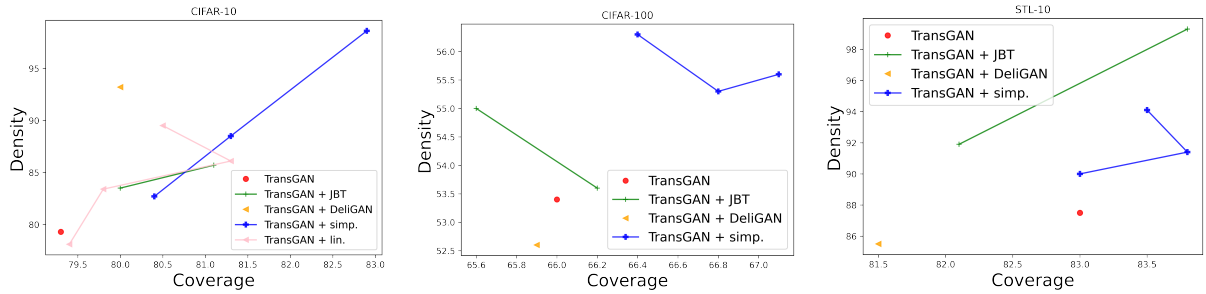


*Figure 8.* Density/Coverage curves comparing TransGAN and boosting methods for multi-modal datasets and different threshold ratios. Our simplicial truncation method (TransGAN + simp.) consistently outperforms the TransGAN and TransGAN + DeliGAN baselines.

| Dataset | Model | FID | Prec | Rec | Dens. | Cov. |
|---|---|---|---|---|---|---|
| *CIFAR-10* | TransGAN | $8.9 \pm 0.1$ | $72.8 \pm 0.8$ | $62.6 \pm 0.7$ | $79.3 \pm 0.9$ | $79.3 \pm 1.2$ |
| | TransGAN + 90% JBT | $8.7 \pm 0.1$ | $73.0 \pm 0.6$ | $61.9 \pm 0.8$ | $83.5 \pm 1.7$ | $80.0 \pm 1.6$ |
| | TransGAN + 80% JBT | $8.8 \pm 0.1$ | $73.3 \pm 0.8$ | $61.2 \pm 1.0$ | $85.7 \pm 2.8$ | $81.1 \pm 0.6$ |
| | TransGAN + DeliGAN N=10 | $9.8 \pm 0.1$ | $74.6 \pm 0.8$ | $58.6 \pm 0.9$ | $93.2 \pm 2.8$ | $80.0 \pm 0.6$ |
| | TransGAN + lin. N=10 (0.23,0.23) | $9.2 \pm 0.1$ | $73.1 \pm 1.1$ | $61.9 \pm 1.2$ | $78.1 \pm 2.7$ | $79.4 \pm 0.9$ |
| | (0.23,0.29) | $9.2 \pm 0.1$ | $73.7 \pm 0.8$ | $61.5 \pm 1.2$ | $83.4 \pm 3.3$ | $79.8 \pm 0.5$ |
| | (0.23,0.31) | $9.3 \pm 0.1$ | $74.0 \pm 0.5$ | $61.0 \pm 0.7$ | $86.1 \pm 1.8$ | $81.3 \pm 0.7$ |
| | (0.23,0.4) | $9.8 \pm 0.1$ | $75.1 \pm 0.8$ | $59.8 \pm 1.2$ | $89.5 \pm 1.6$ | $80.5 \pm 1.2$ |
| | TransGAN + simp. N=10, (0.135,0.135) | $9.0 \pm 0.1$ | $72.9 \pm 0.5$ | $61.8 \pm 0.9$ | $82.7 \pm 1.9$ | $80.4 \pm 0.7$ |
| | (0.135,0.14) | $9.0 \pm 0.1$ | $74.2 \pm 1.5$ | $60.7 \pm 1.0$ | $88.5 \pm 3.1$ | $81.3 \pm 1.4$ |
| | (0.135,0.1445) | $9.3 \pm 0.1$ | $75.3 \pm 0.6$ | $58.8 \pm 0.8$ | $98.6 \pm 1.3$ | $82.9 \pm 0.5$ |
| *CIFAR-100* | TransGAN | $15.2 \pm 0.1$ | $64.2 \pm 0.5$ | $63.1 \pm 0.9$ | $53.4 \pm 1.3$ | $66.0 \pm 1.1$ |
| | TransGAN + 90% JBT | $15.1 \pm 0.2$ | $64.8 \pm 1.0$ | $62.9 \pm 1.3$ | $53.6 \pm 2.4$ | $66.2 \pm 1.4$ |
| | TransGAN + 80% JBT | $14.8 \pm 0.2$ | $65.4 \pm 1.7$ | $61.7 \pm 1.2$ | $55.0 \pm 4.0$ | $65.6 \pm 2.3$ |
| | TransGAN Deligan 10 | $15.9 \pm 0.2$ | $63.5 \pm 0.8$ | $62.2 \pm 0.7$ | $52.6 \pm 1.3$ | $64.4 \pm 0.6$ |
| | TransGAN DeliGAN 100 | $15.3 \pm 0.1$ | $64.2 \pm 0.5$ | $61.9 \pm 0.9$ | $52.6 \pm 0.6$ | $65.9 \pm 0.8$ |
| | TransGAN + simp. N=10, (0.135,0.135) | $15.1 \pm 0.1$ | $65.1 \pm 0.6$ | $62.3 \pm 0.5$ | $55.6 \pm 0.6$ | $67.1 \pm 0.5$ |
| | (0.135, 0.14) | $15.1 \pm 0.1$ | $64.8 \pm 0.2$ | $61.1 \pm 0.5$ | $55.3 \pm 1.3$ | $66.8 \pm 1.1$ |
| | (0.135,0.1445) | $15.1 \pm 0.1$ | $65.6 \pm 1.3$ | $61.5 \pm 0.8$ | $56.3 \pm 1.5$ | $66.4 \pm 1.4$ |
| *STL-10 (32x32)* | TransGAN | $10.5 \pm 0.1$ | $75.7 \pm 0.6$ | $60.1 \pm 0.8$ | $87.5 \pm 1.9$ | $83.0 \pm 0.2$ |
| | TransGAN + 90% JBT | $10.5 \pm 0.1$ | $76.9 \pm 0.7$ | $58.8 \pm 0.5$ | $91.9 \pm 1.9$ | $82.1 \pm 0.8$ |
| | TransGAN + 80% JBT | $11.0 \pm 0.1$ | $78.1 \pm 0.3$ | $57.6 \pm 1.3$ | $99.3 \pm 2.8$ | $83.8 \pm 0.7$ |
| | TransGAN DeliGAN 10 | $12.1 \pm 0.1$ | $74.2 \pm 1.2$ | $60.2 \pm 0.5$ | $81.5 \pm 1.5$ | $79.6 \pm 0.8$ |
| | TransGAN DeliGAN 100 | $10.5 \pm 0.2$ | $76.0 \pm 0.5$ | $60.2 \pm 1.6$ | $85.5 \pm 2.8$ | $81.5 \pm 1.4$ |
| | TransGAN + simp. N=100, (0.08,0.08) | $10.1 \pm 0.1$ | $76.5 \pm 0.9$ | $60.2 \pm 0.8$ | $90.0 \pm 1.7$ | $83.0 \pm 0.5$ |
| | (0.08,0.15) | $10.0 \pm 0.1$ | $76.9 \pm 0.8$ | $59.9 \pm 0.6$ | $91.4 \pm 1.1$ | $83.8 \pm 0.3$ |
| | (0.08,0.20) | $10.0 \pm 0.1$ | $77.8 \pm 0.6$ | $59.8 \pm 0.8$ | $94.1 \pm 0.9$ | $83.5 \pm 0.8$ |

*Table 7.* Density/Coverage curves comparing TransGAN and boosting methods for multi-modal datasets and different threshold ratios. Our simplicial truncation method (TransGAN + simp.) consistently outperforms the TransGAN and TransGAN + DeliGAN baselines.

### B.4. Impact of the number of modes: a synthetic example (Details for Section 4.2)

To illustrate our theoretical results, we propose to vary the number of modes of the data distribution. On real-world data, the number of modes is set but usually unknown, and removing/adding classes as a proxy for modes usually does not give insightful results since some classes can be much more complex than others. We thus use a synthetic setting, where we can easily control both the number of modes and their complexity. Figure 9 stresses that as the number of modes increase, the precision decrease. Interestingly, using large latent space dimension can relieve the problem, even if the latent space dimension is clearly below that of the target. Recall the two problems that arise when training GANs: i) *dimensional misspecification* where the true and modeled distributions do not have density functions w.r.t. the same base measure, and ii) *density misspecification*, where GANs try to fit a disconnected manifold with a unimodal disitribution. From the results we conclude that:

- With very low latent space dimensions, both problems i) and ii) have to be addressed and this leads to poor precision as the number of modes increases.
- With larger latent space dimensions, the problem i) is less of a burden even when there is a clear dimensional misspecification and thus the GANs' performance is more tied to problem ii).

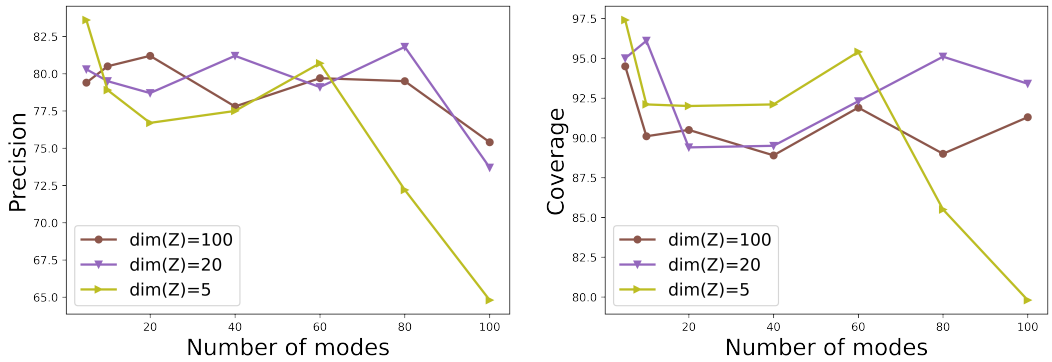


*Figure 9.* Training on a mixture of Gausians in $\mathbb{R}^{100}$ with varying number of modes and varying latent space dimension. The bigger the number of modes, the lower the precision. Increasing the latent space dimension helps up to a limit depending on the number of modes.