Looser is Worse: On the Coding Space of Deep Generative Models

Naren Manoj
Department of Computer Science
The University of Texas at Austin
Austin, TX 78705, USA
manoj.narens@utexas.edu

August 14, 2018

Abstract

In this paper, we identify an undesirable property of the latent space of Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs). We call this property *looseness* and prove that it has negative implications for the diversity of the samples generated from such models. Due to our result, we claim that an efficiently computable proxy for this property implies an efficient test to help compare the sample diversity of two generative models. We also devise and evaluate a regularizer for any VAE training objective that discourages looseness. Although our results do not show substantial improvements in sample and interpolation quality over vanilla VAEs, we believe our intuition will inspire better generative model training algorithms that will more successfully penalize looseness.

1 Introduction

Deep generative models are powerful methods used to represent a distribution over natural images. Two popular instances of deep generative models are Variational Autoencoders (VAEs) (Kingma & Welling (2013)) and Generative Adversarial Networks (GANs) (Goodfellow et al. (2014)).

In this paper, we make two contributions - one theoretical and one empirical. On the theoretical side, we show that one measure of the *looseness* of the latent space of a deep generative model, which we term the *class radius*, can be used to bound a function of the diversity of the generated samples. In particular, a high class radius translates into an upper bound on the diversity of the samples as measured by collision probability. Our analysis uses results obtained by and closely parallels the methods used by Fawzi et al. (2018). On the empirical side, we present a modification to VAE training algorithms that is intended to "fill holes" in the coding space of the VAE; in other words, our method aims to minimize the probability mass of the latent space that is unmapped to by the encoder. This therefore might allow the VAE to better capture the data manifold (Makhzani et al. (2015)). Our method can be applied to any variant of a VAE. Although this method does not have immediate noticeable effects on the sample quality of a vanilla VAE, a more extensive evaluation of this method and of other techniques to learn a better latent space is an important direction for future work.

The rest of this paper is organized as follows. In Section 2, we define important terminology. In Section 3, we state and discuss our theoretical claims relating the class radius to the diversity of a generative model. In Section 4, we motivate and describe our training mod-

ification and present some preliminary results. Finally, in Section 5, we contextualize our work with respect to other efforts in related areas and suggest problems of further interest.

2 Notation and Terms

In this section, we formally define and explain the terms we use throughout the remainder of this paper.

Definition 2.1 (Deep Generative Model)**.** *A* deep generative model *is a (possibly randomized) function, denoted g, that accepts some vector $z \in \mathbb{R}^d$ for some d as input and outputs some image $I \in \mathcal{I}$, where $\mathcal{I}$ represents the space of images.*

Definition 2.2 (Encoder)**.** *An* encoder *is a (possibly randomized) function, denoted f, that accepts some input image $I \in \mathcal{I}$ (where, again $\mathcal{I}$ is the space of images) and outputs some vector $z \in \mathbb{R}^d$.*

State of the art implementations of deep generative models and encoders include Variational Autoencoders[1], Generative Adversarial Networks (GANs)[2][3], and their respective variants. Throughout the remainder of this paper, we use the terms *latent vector* and *coding vector* interchangeably with the input vector for a model g. We call the input space of g and the output space of f the *latent space* or the *coding space*.

Definition 2.3 (Class)**.** *Suppose we are given a classifier into K classes $c : \mathcal{I} \to [K]$ that accepts as input an image and outputs a class label which is an integer index in $[K] \coloneqq \{i\}_{i=1}^{K}$. Then, the* class *of an image I under c is given by:*

$$K_{c(I)} \coloneqq \{z' \mid c\left(g\left(z'\right)\right) = c(I)\}$$

In words, K_i is the subset of the latent space that is classified into label i when a point $z \in K_i$ is passed through g and subsequently classified by c.

Observe that our definition of class is very general. Notably, a class could be the set of latent vectors mapping to a particular classification of an image under some classifier (for instance, K_i for some i could be all the latent vectors mapping to images of dogs). Similarly, a class could be the set of latent vectors mapping to every image at most ϵ away from some reference image in ℓ_p distance.

Furthermore, notice that K_i need not be connected even for simple functions g and classifiers c.

Example 2.1. *Suppose z is distributed according to a standard univariate Gaussian, $g(z) = z^2$, and $c(x) = 1 + \mathbb{1}_{x \geq 1}$. Thus, we have:*

$$K_1 = (-1, 1)$$
$$K_2 = (-\infty, -1] \cup [1, \infty)$$

Note that K_2 is not connected.

In Example 2.2, we discuss a slightly more interesting scenario to help clarify our terminology.

Example 2.2. *Consider a generative model $g : \mathbb{R}^2 \to [-5, 5]^2$. Define c below:*

$$c(p) = \underset{0 \leq i,j \leq 4}{\arg\min} \|(-4 + 2i, -4 + 2j) - p\|_2$$

Thus, $c(p)$ is equivalent to sending p to its nearest lattice point with only even coordinates. For a more concrete visualization of these concepts under this example, see Figure 2.1.

[1] Kingma & Welling (2013)

[2] Goodfellow et al. (2014)

[3] Though a GAN does not come with an encoder by default, numerous GAN flavors involve training an encoder.

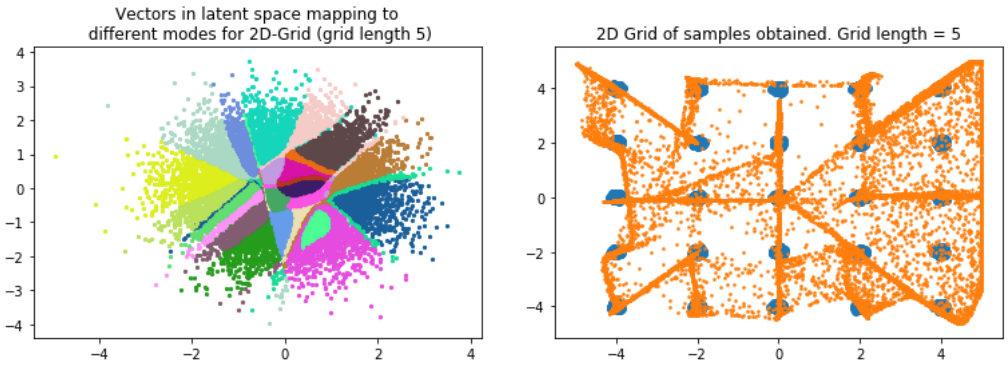

Figure 2.1: Left image: Points sharing the same color are in the same class under the classification function c and generative model g defined in Example 2.2. Right image: Mapping of points in the space depicted in the left image under a generative model g. Each blue dot represents a distinct class, and each orange dot represents a sample obtained by applying g to a vector sampled from the left image. Thus, the classification function c sends each orange dot to its closest blue dot.

The central focus of this paper is addressing "looseness" of the latent or coding space of a generative model. Roughly speaking, the latent space exhibits this undesirable property if there exist many latent vectors that are unmapped to by an encoder encoding the data distribution or if the average distance from a randomly selected latent vector to another latent vector belonging to a different class is high. Notice that both of these ideas are similar - in particular, the first property above often implies the second.

To formalize this idea, we introduce the following definition.

Definition 2.4 (Latent Hole)**.** *A* hole *is a point $z \in \mathbb{R}^d$ such that there does not exist an image I in the data manifold for which $f(I) = z$.*

A latent space is now loose if the volume of the set of holes is high.

Looseness is an undesirable property for several reasons. Makhzani et al. (2015) claim that a model with many holes in the latent space has failed to capture the data manifold as well as another model with fewer holes. This could result in lower quality of generated samples. For a concrete visualization, see Figure 2.2.

We also define another tool to characterize looseness below.

Definition 2.5 (Class Radius)**.** *The* class radius *is the expected distance from a random point $z \sim \mathcal{N}\left(0, \text{diag}\left(\vec{1}_d\right)\right)$ in the latent space to the nearest class boundary. Specifically, define $r(z)$ below:*

$$r(z) \coloneqq \inf_{z' \in \mathcal{Z}/K_{c(g(z))}} \|z - z'\|$$

Then the class radius is simply:

$$\mathop{\mathbb{E}}_{z \sim \mathcal{N}\left(0, \text{diag}\left(\vec{1}_d\right)\right)} [r(z)]$$

Notice that the existence of many holes implies that $r(z)$ must be high for many points z, which increases the class radius. Thus, roughly speaking, a loose latent space also has a high class radius. A high class radius is undesirable because, as we assert in Section 3, this yields an upper bound on the diversity of the generated distribution. Hence, since we usually want to train generative models capable of generating more diverse samples, we therefore prefer generative models with lower class radii.

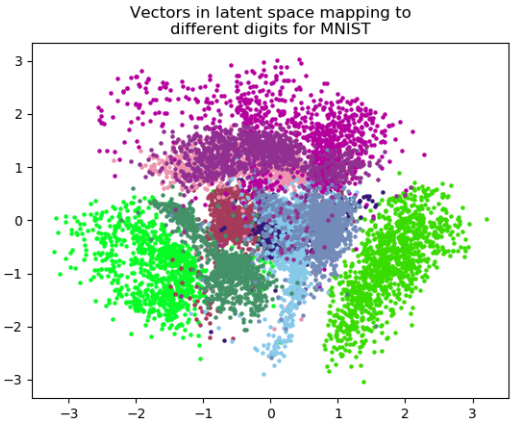

Figure 2.2: Latent space of a VAE trained on MNIST with latent dimension of 2. Observe the large gaps between classes; these large gaps are holes in the latent space. Thus, an interpolation between two classes separated by a hole would result in the generated images "falling off" the data manifold.

3 Theoretical: A High Class Radius is Bad for Diversity

In this section, we state a bound directly relating the *collision entropy* of the generated distribution to the class radius. Thus, *the class radius can be bounded by a function of the diversity of the generative model.* Under a reasonable probability decay assumption (in particular, the maximum probability of any class is a decreasing function in K, the number of classes), we show that our upper bound is a decreasing function in the number of classes K. Thus, by a simple squeeze argument, we have that the class radius decays to 0 as the number of classes grows arbitrarily large. Additionally, we show that our bound is smallest when the generator generates a uniform distribution across its K modes. This may imply that a generative model has learned a potentially more useful distribution; we provide some basic experiments to back up this idea.

The rest of this section is organized as follows. We first formally state and define the problem setting under which we work. Next, we state our central claims. Finally, we discuss some definite and potential practical implications of our results.

Notation and Problem Setting Suppose that we are given access to a *deterministic* generative model g that accepts as input some vector $z \sim \mathcal{N}\left(0, \text{diag}\left(\vec{1}_d\right)\right)$ and outputs some image $I \in \mathcal{I}$, where d is the dimension of the latent space and $\mathcal{I}$ is the space of images. Furthermore, suppose we have a classification function c with K possible outputs. Let the probability that an image of class K_i is generated be p_i, and let the vector of these probabilities be p. Let $\varphi(x)$ denote the standard univariate Gaussian probability density function (PDF), $\Phi(x)$ denote the standard univariate Gaussian cumulative distribution function (CDF), and $\Phi^{-1}(x)$ denote the inverse of $\Phi(x)$.

For the purpose of our proofs, we enforce the following reasonable assumption on the generated distribution:

$$\|p\|_\infty \leq \min\left(\frac{1}{5}, h(K)\right) \text{ such that } h(K) = \Omega\left(\frac{1}{K}\right) \text{ and } h(K) = o(1)$$

Notice that this just means that the maximum probability of any class does not remain constant; instead, the maximum probability decays as the number of classes K increases. One way to view this constraint is to observe that we are enforcing some kind of balance on the classes. By limiting the maximum probability of any class, we are ensuring that no class becomes too asymptotically dominant.

Now, we state our central theoretical result. Informally, we can upper bound the class radius based on the distribution induced by our generative model.

Theorem 3.1. *If $0 < p_i \leq 1/5, \forall i \in [K]$, then the following holds regarding the expected class radius of each image in the latent space of our generative model:*

$$\mathbb{E}\left[r(z)\right] \leq \frac{\log\left(4\pi \log\left(1/\left\|p\right\|_2^2\right)\right)}{\sqrt{2\log\left(1/\left\|p\right\|_2^2\right)}}$$

Proof. We defer a full proof to the appendices; see the proof of Theorem B.1. □

Observe that $\left\|p\right\|_2^2$ is exactly the following:

$$\mathop{\mathbb{P}}_{z_1, z_2 \sim \mathcal{N}\left(0, \mathrm{diag}\left(\vec{1}_d\right)\right)}\left[f\left(g\left(z_1\right)\right) = f\left(g\left(z_2\right)\right)\right]$$

Furthermore, recall that $-\log\left(\left\|p\right\|_2^2\right)$ is the second-order Renyi entropy, which is also known as the collision entropy[4]. Thus, our upper bound on the class radius can also be viewed as a measure of the diversity of the generated distribution, which supports the use of the class radius as a measure of diversity.

More concretely, observe that the function $\log\left(4\pi x\right)/\sqrt{2x}$ is decreasing in x. Thus, inverting the bound in Theorem 3.1 yields an upper bound on $-\log\left(\left\|p\right\|_2^2\right)$. As a result, having access to $\mathbb{E}\left[r(z)\right]$ for some generative model g and classification function f translates into an upper bound on the diversity of g with respect to f.

We can also show that to minimize the bound in Theorem 3.1, we must take $p_i = 1/K$. We state this in the following theorem.

Theorem 3.2. *The upper bound stated in Theorem 3.1 is minimized when $p_i = 1/K$. Informally, if the generative model induces the uniform distribution over possible images, then the expected class radius is smallest. The bound in Theorem 3.1 then becomes:*

$$\mathbb{E}\left[r(z)\right] \leq \frac{\log\left(4\pi \log\left(K\right)\right)}{\sqrt{2\log\left(K\right)}}$$

Proof. We defer a full proof to the appendices; see the proof of Theorem B.2. □

We now show that the bound in Theorem 3.1 is decreasing in K. Thus, as K grows, we must have that $\mathbb{E}\left[r(z)\right]$ approaches 0. We formalize and prove this claim below.

Theorem 3.3. *Suppose that we impose the following constraint on p_i:*

$$p_i \in \left[0, \min\left(\frac{1}{5}, h(K)\right)\right]$$

where $h(K)$ is a function of K such that $h(K) = o(1)$ and $h(K) = \Omega(1/K)$. Then, we have:

$$\lim_{K \to \infty} \mathbb{E}\left[r(z)\right] = 0$$

Proof. We defer a full proof to the appendices; see the proof of Theorem B.3. □

[4]Recall that the α-order Renyi entropy is $\alpha\left(1-\alpha\right)^{-1}\log\left(\left\|p\right\|_\alpha\right)$. Taking the limit as α approaches 1 yields the Shannon entropy, which is simply $-\sum p_i \log\left(p_i\right)$. We obtain the collision entropy by setting $\alpha = 2$.

3.1 Theory in Practice

We now propose several ways to leverage these observations in practice. One interesting problem is to determine a test statistic that can be computed given a generator g and a classifier f. This test statistic can then be used to make claims regarding the diversity of the generative models. In particular, if we could easily compute $\mathbb{E}[r(z)]$, then we could compare two generative models. The theorems above suggest that a higher value of $\mathbb{E}[r(z)]$ implies that g is less diverse with respect to f (recall that our upper bounds decrease as the generated distribution approaches uniformity and as the number of outputs K grows arbitrarily large). Unfortunately, it is not immediately obvious how to compute $\mathbb{E}[r(z)]$ more efficiently than measuring the collision probability directly. Presenting an easily computable proxy for $\mathbb{E}[r(z)]$ would therefore be incredibly useful, as we believe that this would imply a potentially useful test of diversity for a generative model.

As a first step towards demonstrating the viability of such a test, we approximate $\mathbb{E}[r(z)]$ for a toy example. We train multiple GANs on variants of the 2D-Grid synthetic dataset[5] (Lin et al. (2017); Srivastava et al. (2017)) and observe various degrees of mode collapse on each. We use this simple synthetic dataset because measuring mode collapse is easy when the modes are known and because the results of this task are easy to visualize. The results are shown in Figure 3.1. It is clear from this figure that the GANs capturing more modes enjoy a lower class radius value, which indicates that obtaining an easily computable approximation to $\mathbb{E}[r(z)]$ would likely be useful in comparing the diversity of two generative models.

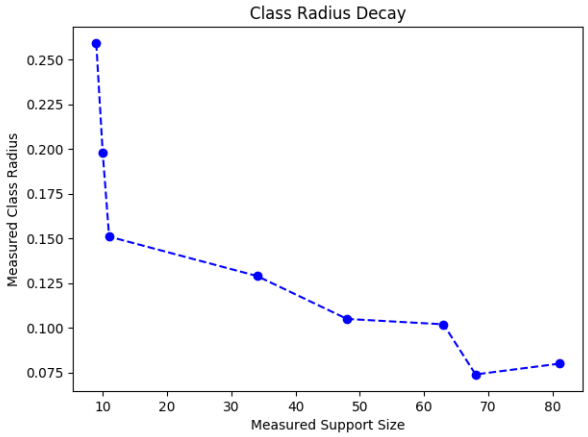

Figure 3.1: In the above figure, we plot the (discretized) support size of a GAN trained on the 2D-Grid task against the approximation of the class radius we compute.

Another problem of interest is to use $\mathbb{E}[r(z)]$ or some proxy to $\mathbb{E}[r(z)]$ as a regularizer for training a generative model. Such a method should penalize a lack of diversity and should therefore encourage the model to learn a better distribution. In Section 4, we describe one effort towards modifying the objective of a VAE to achieve this desired effect. However, there remain many unexplored directions to pursue.

3.2 Connection to Adversarial Robustness

In this section, we point out a tangential implication of our theoretical results. Our proofs of the theorems in the preceding sections are heavily inspired by those by Fawzi et al. (2018),

[5]The 2D-Grid task consists of a target distribution that is a mixture of $(2n+1)^2$ spherical two-dimensional Gaussians with variances 0.0025 and means $(2i, 2j), -n \leq i, j \leq n$. The classification function c we use sends each generated point to its nearest mean, as done in the special case in Example 2.2.

as their problem setting is equivalent to ours. Fawzi et al. (2018) consider the problem of determining robustness against *adversarial examples*. Adversarial examples are perturbed inputs to a classifier with the intention of changing the classifier outputs. The adversarial input is usually nearly identical to the original input. Due to the equivalence of the problem settings, we can apply our results to the adversarial examples domain.

Perhaps the biggest implication of our results to adversarial examples is that the expected in-distribution robustness is smallest when each class is equiprobable, a result of Theorem 3.2. Furthermore, as a result of Theorem 3.3, we have that the expected in-distibution robustness decays to 0 as long as the probability of the most probable class decays with the number of classes K. This provides some theoretical basis for the results obtained by Jordan et al. (2018) in which the authors observe that far smaller perturbations are required to adversarially attack an ImageNet classifier than are required to attack a CIFAR-10 classifier. Since ImageNet contains far more classes than CIFAR-10 and no class in either dataset is vastly overrepresented, the conditions required for our theorems to apply are met.

4 Experimental: Lipschitz-Constrained VAE

In this section, we present our modification to the encoder of the VAE. Our modification involves constraining the encoder to have a low Lipschitz constant, which helps prevent large gaps in the coding space from existing. To implement the Lipschitz constraint, we follow Arjovsky et al. (2017) and constrain the weights of the encoder to lie in $[-c, c]$ for some hyperparameter c. We also add a regularizer penalizing the distances between the encodings weighted by the distance between the original inputs. We train various VAEs with our modifications and demonstrate the results here. We propose naming our modification *LC-VAE* (standing for Lipschitz Constrained Variational Autoencoder).

4.1 Motivation

Recall that we would like all our encodings to lie close to one another. This way, we minimize the number of "holes" in the coding space and we can minimize the class radius, thereby helping prevent a lack of diversity in generated samples. Mathematically, if we let f be our encoder, then we would like to make the following quantity small over all input images I_1, I_2:

$$\|f(I_1) - f(I_2)\|_2$$

Thus, one way to penalize distant encodings would be to compute the above quantity over many pairs of images and use it as a regularizer alongside the vanilla loss function used by the VAE. However, notice that this does not take into account the distance between the original input images. For instance, suppose we were computing encodings for the MNIST dataset. If we used this regularization as-is, then we would be penalizing the distance between the encodings of, say, a "1" and a "9" as much as we would be for two different "1"s. This is clearly not desired, and we would therefore like to weight our penalization by the distance between the original images. In particular, if the two input images are close to one another, then we care more that their encodings are close than if the input images are distant.

Therefore, we could regularize the VAE objective using the following quantity over many pairs of images I_1, I_2:

$$\omega\left(I_1, I_2\right) \coloneqq \frac{\|f(I_1) - f(I_2)\|_2^2}{\|I_1 - I_2\|_2^2}$$

Observe that the above quantity is upper-bounded by the Lipschitz constant of f. Thus, if we would like this quantity to be generally small, we can further constrain the regularization by enforcing f to be L-Lipschitz for some L. To force f to have some Lipschitz constant, we additionally parameterize f by forcing its weights to lie within $[-c, c]$ for some positive constant c. It is well-known that weight clipping is not a good way to enforce a Lipschitz constant (Arjovsky et al. (2017)), and a better way to penalize a high Lipschitz constant of f remains an open problem.

Our final modifications involve regularizing the VAE objective using ω with weight λ and clamping the weights of f with some constant c. This yields two additional hyperparameters

to tune (λ and c). Although our modifications can be applied to any VAE flavor, we experiment with the vanilla VAE in this paper.

Implementation Details For speed, we compute the following function as our regularizer for a given batch when the batch size is $b = 2n$ for some integer n:

$$\sum_{i=1}^{n} \omega\left(I_i, I_{i+n}\right)$$

4.2 Experimental Results

We evaluate LC-VAE on the MNIST dataset (LeCun & Cortes (2010)). We implement LC-VAE in PyTorch 0.4.1 and use a slightly modified version of the VAE from the PyTorch examples repository.

Experimental Setup All our experiments use a slightly modified VAE architecture. Our encoder's convolutional architecture is described Table 1. To generate our vectors of means and standard deviations, we follow the network described in Table 1 with a single fully-connected layer with 1024 inputs and d outputs, where d is the latent dimension. Our decoder's convolutional architecture is described in Table 2. Note that each convolutional layer is a transposed convolution; furthermore, we precede the layers in Table 2 with a fully-connected layer with d inputs and $12544 = 256 \times 7 \times 7$ outputs.

	Input Channels	Output Channels	Kernel Size	Stride	Padding	Batch Norm?	Activation
1	1	128	4	2	`SAME`	Yes	ReLU
2	128	256	4	2	`SAME`	Yes	ReLU
3	256	512	4	2	`SAME`	Yes	ReLU
4	512	1024	4	2	`SAME`	Yes	ReLU

Table 1: Encoder convolutional architecture.

	Input Channels	Output Channels	Kernel Size	Stride	Padding	Batch Norm?	Activation
1	256	128	4	2	`SAME`	Yes	ReLU
2	128	1	4	2	`SAME`	No	Sigmoid

Table 2: Decoder (transposed) convolutional architecture.

We train each network for 20 epochs using a batch size of 300. We use the Adam optimizer (Kingma & Ba (2014)) with learning rate 0.001. Unless otherwise specified, we use a latent dimension of size 10. We set $\lambda = 30$ and $c = 0.30$ as our parameters for our regularizer and Lipschitz constraint, respectively.

LC-VAE Tightens Latent Space As shown in Figure 4.1, LC-VAE indeed prevents looseness of the latent space. Observe that the vanilla VAE leaves far larger holes in the latent space than does LC-VAE; furthermore. Thus, an interpolation performed by drawing a line between the two latent representations of two images spends less time off the data manifold when the latent representations are determined via our modifications.

The structure of the latent space of LC-VAE comes with the additional benefit of generating more diverse samples. In particular, the probability that the dark brown class is generated is higher for LC-VAE since this region is closer to the origin than the dark brown class for the vanilla VAE. The same is true for the beige class and the bright blue class. However, the primary downside to this structure is that the probability that a latent vector does not belong to any of these classes (and therefore potentially generates a nonsense image) is higher for LC-VAE than it is for the vanilla VAE.

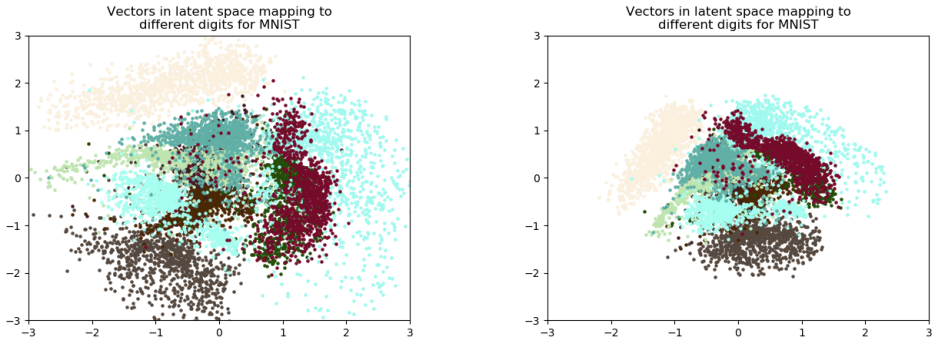

Figure 4.1: Left image: Visualization of the two-dimensional latent space of a vanilla VAE. Right image: Visualization of the two-dimensional latent space of LC-VAE. Excluding the latent space, all hyperparameters remained the same as previously specified.

LC-VAE Does Not Impact Sample Quality Figures 4.2 and 4.3 depict samples and reconstructions, respectively, from VAE and LC-VAE. Observe that the sample quality is not obviously impacted by the addition of our regularizer. Thus, constraining the geometry of the latent space does not by itself affect image quality.

The lack of change in image quality might be due to the encoder "cheating" the regularizer. In particular, there might be some scaling effect happening to the latent space since only the KL term in the loss prevents the network from just reproducing a geometrically similar latent space to the one it would have learned without the regularization. Thus, although some holes are filled by LC-VAE, due to this scaling effect, there is not much (if any) improvement in sample quality.

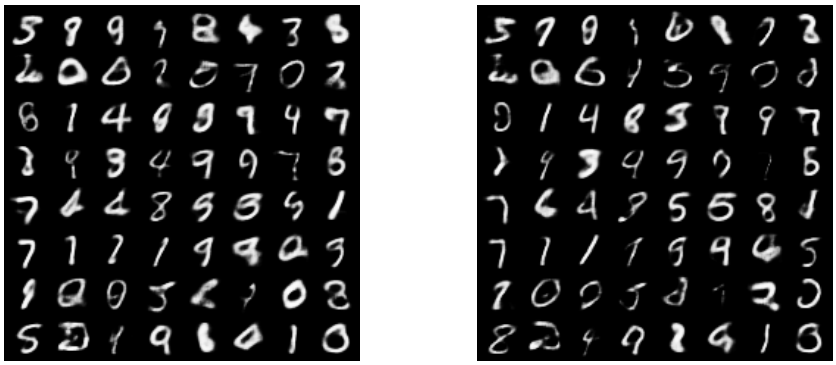

Figure 4.2: Left image: Samples from vanilla VAE with latent dimension 10. Right image: Samples from LC-VAE with latent dimension 10. The latent code is sampled from $\mathcal{N}\left(0, \text{diag}\left(\vec{1}_d\right)\right)$.

Figure 4.3: Left image: Reconstructions from vanilla VAE with latent dimension 10. Right image: Reconstructions from LC-VAE with latent dimension 10. Original inputs are shown on the top row and reconstructions are shown on the bottom row.

Future Work A more extensive experimental evaluation of LC-VAE remains for future work. There may be some combination of hyperparameters that we did not try that may yield better results. Furthermore, MNIST is just a toy dataset. Thus, to more widely observe the impact of our regularizer, we would need to evaluate LC-VAE's quality on more difficult datasets such as CelebA (Liu et al. (2015)). Furthermore, it would be interesting to see how our modifications impact the performance of other flavors of the VAE.

5 Conclusion

In this paper, we identified a property of the latent space of generative models called *looseness*. We discussed its negative implications regarding the modelling of the data manifold and proved that a high class radius implies an upper bound on the diversity of the generative model. We then proposed a few directions which leverage our theoretical insights for practical purposes. Finally, we evaluated one possible training modification to VAEs discouraging looseness. Although our initial results do not indicate significant improvements over existing VAEs, we leave further exploration of these ideas for future work.

Future Work As previously mentioned, our work leaves open the problem of leveraging our theoretical observations to improve the training of generative models and to potentially devise additional techniques to evaluate existing generative models. Additionally, it remains an open problem to design a better regularizer for the vanilla VAE objective that both penalizes looseness and results in the VAE learning the data manifold better. A more broad issue of interest is developing a more theoretical understanding of generative models, which we hope our work contributes towards and sparks further research in.

Related Work On the theoretical side, the paper by Fawzi et al. (2018) is most similar to ours. Our bounds are based on results the authors obtain for the adversarial examples domain and our analysis is heavily inspired by theirs. In this paper, the authors prove that under some assumptions, every classifier is vulnerable to small adversarial perturbations. One key assumption they make is that the data distribution on which the classifier operates comes from a generative model whose latent vectors are distributed according to a spherical Gaussian with identity covariance. This assumption is commonly reflected in generative models such as VAEs and GANs and their respective variants.

On the empirical side, there are various papers that make similar contributions to ours. Arora et al. (2018) attack the problem of determining diversity of GANs by leveraging the birthday paradox. Their so-called birthday paradox test hinges on the fact that high collision probabilities are evidence of low diversity. This provides another angle from which to view our theoretical result relating collision probabilities to the class radius. Additionally, this work provides one solution to a problem we pose - namely, producing a test that can be used to evaluate the diversity of a generative model. Borji (2018) describes a zoo of existing methods to evaluate generative models, and Khrulkov & Oseledets (2018) and Santurkar et al. (2017) present more recent methods to evaluate generative models. However, except for Khrulkov & Oseledets (2018), no authors of recent generative model evaluation strategies leverage geometric insights.

References

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

Sanjeev Arora, Andrej Risteski, and Yi Zhang. Do GANs learn the distribution? some theory and empirics. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=BJehNfW0-`.

Ali Borji. Pros and cons of gan evaluation measures. *arXiv preprint arXiv:1802.03446*, 2018.

Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

Alhussein Fawzi, Hamza Fawzi, and Omar Fawzi. Adversarial vulnerability for any classifier. *arXiv preprint arXiv:1802.08686*, 2018.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

He Huang, Phillip Yu, and Changhu Wang. An introduction to image synthesis with generative adversarial nets. *arXiv preprint arXiv:1803.04469*, 2018.

Matt Jordan, Naren Manoj, Surbhi Goel, and Alex Dimakis. Combined adversarial attacks. *NIPS 2018 (submitted)*, 2018.

Valentin Khrulkov and Ivan Oseledets. Geometry score: A method for comparing generative adversarial networks. *arXiv preprint arXiv:1802.02664*, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/, 2010. URL `http://yann.lecun.com/exdb/mnist/`.

Fei-Fei Li, Justin Johnson, and Serena Yeung. Lecture notes for convolutional neural networks for visual recognition, May 2018.

Zinan Lin, Ashish Khetan, Giulia Fanti, and Sewoong Oh. Pacgan: The power of two samples in generative adversarial networks. *arXiv preprint arXiv:1712.04086*, 2017.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pp. 2234–2242, 2016.

Shibani Santurkar, Ludwig Schmidt, and Aleksander Madry. A classification-based perspective on gan distributions. *arXiv preprint arXiv:1711.00970*, 2017.

Akash Srivastava, Lazar Valkoz, Chris Russell, Michael U Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. In *Advances in Neural Information Processing Systems*, pp. 3308–3318, 2017.

A Deep Generative Models: A Whirlwind Tour

In this section, we give a brief background on the deep generative models we consider in this paper.

A *generative model* solves the following problem (Li et al. (2018)).

Given training data, generate samples from the same distribution as the training data.

A.1 Variational Autoencoders

In this section, we provide a brief overview of Variational Autoencoders. For a more thorough survey, see the work by Doersch (2016). Most of the material in this subsection is borrowed from this survey.

A Variational Autoencoder (VAE) consists of two components. The first is an encoder, which is a potentially nondeterministic function sending inputs to latent codes. The second is a decoder, which is also a potentially nondeterministic function sending latent codes to outputs. The vanilla VAE objective minimizes two terms. The first term is a discrepancy measure between the distribution of encoder outputs and some prespecified distribution over latent codes. The second term is a reconstruction error measuring, essentially, how lossy the compression is that the encoder performs.

In particular, let $P(Z|X)$ be the distribution of encoder outputs conditioned on encoder inputs. In our use case, X is a random vector representing input images distributed according to the data distribution, and Z is a random vector representing the output of the encoder. Let $Q(X|Z)$ be the distribution of decoder outputs conditioned on decoder inputs. Here, X is a random vector denoting the output images of the decoder. If $\mathcal{D}$ is our discrepancy measure between distributions, $\mathcal{R}$ is our discrepancy measure between images, and if we want our latent codes to be distributed according to a distribution $\mathcal{Z}$, then our objective is to minimize:

$$\mathcal{D}\left(P(Z|X), \mathcal{Z}\right) + \mathcal{R}_{\hat{x}\sim Q(X|Z\sim P(Z|X=x))}\left(\hat{x}, x\right)$$

In our experiments, we take $\mathcal{D}$ to be the Kullback-Leibler divergence, $\mathcal{Z}$ to be a spherical Gaussian with identity covariance, and $\mathcal{R}$ to be the ℓ_2 norm between two images.

A.2 Generative Adversarial Networks

In this section, we provide a brief overview of Generative Adversarial Networks. For a more thorough survey, see the work by Huang et al. (2018). Most of the material in this subsection is borrowed from this survey.

A Generative Adversarial Network (GAN) is a deep learning algorithm consisting of two neural networks - a *generator*, which generates images as a function of latent vectors sampled from some distribution $\mathcal{Z}$, and a *discriminator*, which classifies input images as coming from the data distribution $\mathcal{X}$ or from the distribution specified by the generator (given by $G(\mathcal{Z})$, the pushforward of $\mathcal{Z}$ through G).

Algorithm A.1 describes how a GAN is trained. At each step, the generator tries to fool the discriminator by generating realistic images. Thus, the objective of the generator is to maximize the misclassifications that the discriminator makes, and the objective of the discriminator is to maximize its accuracy. This therefore yields the following objective:

$$\min_G \max_D V(D, G) \coloneqq \mathop{\mathbb{E}}_{X\sim\mathcal{X}}\left[\log\left(D(X)\right)\right] + \mathop{\mathbb{E}}_{z\sim\mathcal{Z}}\left[\log\left(1 - D(G(z))\right)\right]$$

GANs have the benefit of generating very sharp images. However, GANs suffer from a problem known as *mode collapse*, in which the generator effectively fails to produce samples from numerous modes in $\mathcal{X}$. Several training modifications have been proposed to mitigate mode collapse (Lin et al. (2017); Salimans et al. (2016); Arjovsky et al. (2017)). On the other hand, researchers have devised methods to detect mode collapse (Arora et al. (2018)), and, more broadly, evaluate the extent to which the GAN has learned the original data manifold (Khrulkov & Oseledets (2018)).

Algorithm A.1 GAN training

Input: Data distribution to model $\mathcal{X}$
Initialize generator G and discriminator D
while G not converged **do**
    Obtain latent vector z from some distribution
    Obtain $X_f = G(z)$ (generated outputs)
    Obtain X_r (real training data)
    Train discriminator by minimizing $\log(D(x_r)) + \log(1 - D(x_f))$
    Train generator by maximizing $\log(D(x_f))$
end while
Output: G, D

B Proofs

In this section, we restate and prove our main results. We begin by showing that our problem setting is equivalent to the one considered by Fawzi et al. (2018). Thus, the bounds the authors obtain are applicable to the problem we consider. We then state and prove intermediate lemmas we employ in our proofs of our main claims. Finally, we state and prove our main theorems.

Equivalence to Fawzi et al. (2018) We show that the results obtained by Fawzi et al. (2018) apply to our problem setting. First, notice that the definition of *in-distribution robustness* is equivalent to Definition 2.5. Next, observe that the assumptions on our classification function match the assumptions on the discriminator detailed by Fawzi et al. (2018). Furthermore, note that the distribution on the latent vectors in both settings is the same. These summarize the assumptions and definitions we make and are equal to those made by Fawzi et al. (2018). Thus, the results the authors obtain are applicable in our setting as well, and we may proceed.

B.1 Lemmas

We first state bounds without proof derived in Fawzi et al. (2018). These bounds form the foundation of our results.

Lemma B.1. *Define:*

$$a_{\neq i} \coloneqq \Phi^{-1}\left(\mathbb{P}\left[\bigcup_{j \neq i} K_j\right]\right)$$

where K_j represents the set of points in latent space mapping to class K_j. Then, the below inequality follows:

$$\mathbb{E}\left[r(z)\right] \leq \sum_{i=1}^{K} -a_{\neq i}\Phi(-a_{\neq i}) + \frac{e^{-a_{\neq i}^2/2}}{\sqrt{2\pi}}$$

Lemma B.2. *If $K \geq 5$, then:*

$$\Phi^{-1}\left(1 - \frac{1}{K}\right) \geq \sqrt{\log\left(\frac{K^2}{4\pi \log(K)}\right)}$$

We also state the following three important equalities regarding the standard Gaussian PDF, CDF, and inverse CDF, respectively.

Lemma B.3. *If we define:*

$$\varphi(x) \coloneqq \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \tag{B.1}$$

$$\Phi(x) \coloneqq \mathop{\mathbb{P}}_{t \sim \mathcal{N}(0,1)}\left[t \leq x\right] = \int_{-\infty}^{x} \varphi(t)\,\mathrm{d}t \tag{B.2}$$

Then the following hold:

$$\varphi(x) = \varphi(-x) \tag{B.3}$$
$$\Phi(-x) = 1 - \Phi(x) \tag{B.4}$$
$$\Phi^{-1}(1-x) = -\Phi^{-1}(x) \tag{B.5}$$

Proof. As a result of Equation B.1, observe that Equation B.3 immediately follows. To see why Equation B.2 holds, note the following for positive x:

$$\mathbb{P}[t \geq x] = \mathbb{P}[t \leq -x] = \Phi(-x)$$

Furthermore, observe the following:

$$\mathbb{P}[t \leq x] = 1 - \mathbb{P}[t \geq x] = \Phi(x)$$

Substituting yields:

$$\Phi(x) + \Phi(-x) = 1$$

as desired. For Equation B.5, notice the following manipulations:

$$\begin{aligned} x &=: \Phi(y) \\ 1 - x &= 1 - \Phi(y) \\ &= \Phi(-y) \\ \Phi^{-1}(1-x) &= -y \\ &= -\Phi^{-1}(x) \end{aligned}$$

as desired. □

Using these, we derive the following lemma, whose notation is largely borrowed from Fawzi et al. (2018).

Lemma B.4. *We can rewrite the upper bound derived from Lemma B.1 as follows:*

$$\sum_{i=1}^{K} -a_{\neq i}\Phi(-a_{\neq i}) + \frac{e^{-a_{\neq i}^2/2}}{\sqrt{2\pi}} = \sum_{i=1}^{K} \Phi^{-1}(p_i) \cdot p_i + \frac{e^{-\Phi^{-1}(p_i)^2/2}}{\sqrt{2\pi}}$$

Proof. We first simplify the following a little:

$$\begin{aligned} \Phi(-a_{\neq i}) &= 1 - \Phi(a_{\neq i}) \\ &= 1 - \Phi\left(\Phi^{-1}\left(\mathbb{P}\left[\bigcup_{j \neq i} K_j\right]\right)\right) \\ &= 1 - \mathbb{P}\left[\bigcup_{j \neq i} K_j\right] \\ &= \mathbb{P}[K_i] \\ &=: p_i \end{aligned}$$

Now, observe the following equality:

$$a_{\neq i} = \Phi^{-1}(1 - p_i) = -\Phi^{-1}(p_i)$$

We use this to put our target expression exclusively in terms of our probability distribution:

$$f(p) \coloneqq \sum_{i=1}^{K} \Phi^{-1}(p_i) \cdot p_i + \frac{e^{-\Phi^{-1}(p_i)^2/2}}{\sqrt{2\pi}}$$

as desired. □

We now revisit the assumption we impose on the vector of generated probabilities p. Under this assumption, we have the following.

Lemma B.5. *Consider the following constraints on p:*

$$\|p\|_\infty \leq v = \min\left(\frac{1}{5}, h(K)\right) \text{ such that } h(K) = \Omega\left(\frac{1}{K}\right) \text{ and } h(K) = o(1)$$
$$\|p\|_1 = 1$$

Then, we have:

$$\frac{1}{K} \leq \|p\|_2^2 \leq \min\left(\frac{1}{5}, 3h(K)\right)$$

Proof. We begin with the lower bound. Notice that the following must hold for any vector $p \in \mathbb{R}^K$:

$$\|p\|_1^2 \cdot \frac{1}{K} \leq \|p\|_2^2$$

Setting $\|p\|_1 = 1$ immediately gives the desired.

We now proceed with the upper bound. Consider the following instantiation of p:

$$p_i = v, \forall i \in \left\{1, \ldots, \left\lfloor \frac{1}{v} \right\rfloor\right\}$$
$$p_{\lfloor 1/v \rfloor + 1} = 1 - v\left\lfloor \frac{1}{v} \right\rfloor$$
$$p_i = 0 \text{ otherwise}$$

Observe that this instantiation of p satisfies both constraints we imposed earlier. Now, consider the following manipulations:

$$\begin{aligned}
\|p\|_2^2 &= \sum p_i^2 \\
&= \left\lfloor \frac{1}{v} \right\rfloor v^2 + \left(1 - v\left\lfloor \frac{1}{v} \right\rfloor\right)^2 \\
&= \left\lfloor \frac{1}{v} \right\rfloor v^2 + \left(1 - 2v\left\lfloor \frac{1}{v} \right\rfloor + v^2\left\lfloor \frac{1}{v} \right\rfloor^2\right) \\
&\leq \frac{1}{v} \cdot v^2 + 1 - 2v\left\lfloor \frac{1}{v} \right\rfloor + v^2 \cdot \frac{1}{v^2} \\
&= v + 2 - 2v\left\lfloor \frac{1}{v} \right\rfloor \\
&\leq v + 2 - 2v\left(\frac{1}{v} - 1\right) \\
&= 3v
\end{aligned}$$

We now show that this is an optimal assignment. Observe that this allocation of mass follows a greedy strategy - we allocate as much mass as we can into each bucket until we run out. To see that the greedy strategy works, note that at each step, we can either allocate as much mass as we can to a bucket, or spread that same mass over several other buckets. To show that allocating as much mass as we can into one bucket alone and leaving the rest empty is optimal, it suffices to show the following for any vector x:

$$\|x\|_1^2 = \left(\sum x_i\right)^2 \geq \|x\|_2^2 = \sum x_i^2$$

This is clearly true, and we may conclude. □

We now state various intermediate results we use to prove our main claims.

Lemma B.6. *The following function is concave if $x \in (0, 1)$:*

$$g(x) \coloneqq \frac{\log\left(4\pi \log\left(1/x\right)\right)}{\sqrt{2\log\left(1/x\right)}}$$

Proof. One sufficient condition for g to be concave is that its second derivative, written below, is always nonpositive on the same domain:

$$g''(x) = \frac{-2\log(1/x)\left(\log\left(4\pi\log(1/x)\right) - 2\right) + 3\log\left(4\pi\log(1/x)\right) - 8}{4\sqrt{2}x^2\left(\log(1/x)\right)^{5/2}}$$

Note that the denominator is always positive on this domain, so it suffices to show that the numerator is always nonpositive on the domain. To clean up notation, make the following substitution:

$$y = \log\left(\frac{1}{x}\right)$$

Then, note that $y \geq 0$. We now wish to show that the below expression is always nonpositive for all nonnegative y:

$$\begin{aligned}
&-2y\left(\log(4\pi y) - 2\right) + 3\log(4\pi y) - 8 \\
&= -2y\left(\log\left(\frac{4\pi}{e^2}\right) + \log(y)\right) + \log\left(\frac{64\pi^3}{e^8}\right) + 3\log(y)
\end{aligned}$$

Notice that as a result of convexity, the following two inequalities hold:

$$\begin{aligned}
-x\log(x) &\leq -x + 1 \\
\log(x) &\leq x - 1
\end{aligned}$$

We use these to our advantage:

$$\begin{aligned}
&-2y\left(\log\left(\frac{4\pi}{e^2}\right) + \log(y)\right) + \log\left(\frac{64\pi^3}{e^8}\right) + 3\log(y) \\
&\leq -2y\left(\log\left(\frac{4\pi}{e^2}\right) + 1\right) + \log\left(\frac{64\pi^3}{e^6}\right) + 3\log(y) \\
&\leq -2y\left(\log\left(\frac{4\pi}{e^2}\right) + 1\right) + \log\left(\frac{64\pi^3}{e^6}\right) + 3(y-1) \\
&= \log\left(\frac{e^5}{16\pi^2}\right)y + \log\left(\frac{64\pi^3}{e^9}\right)
\end{aligned}$$

This is simply a linear equation; thus, to check that it is nonpositive over $y \geq 0$, it is enough to check that the slope and vertical intercepts are nonpositive. This is clearly true here, and we may therefore conclude. □

Lemma B.7. *The following function is increasing when $x \in (0, 1/2]$:*

$$g(x) \coloneqq \frac{\log\left(4\pi\log(1/x)\right)}{\sqrt{2\log(1/x)}}$$

Proof. One sufficient condition for g to be increasing on this domain is that its first derivative, written below, is always positive on the same domain:

$$g'(x) = \frac{\log\left(4\pi\log(1/x)\right) - 2}{2\sqrt{2}x\left(\log(1/x)\right)^{3/2}}$$

Note that the denominator is always positive on this domain, so it suffices to show that the numerator is always positive on the domain. Specifically, we wish to show that for $x \in (0, 1/2]$:

$$\log\left(4\pi\log\left(\frac{1}{x}\right)\right) > 2$$

Since the LHS is always increasing in $1/x$, it follows that to minimize the LHS, we must minimize $1/x$. This occurs by setting $x = 1/2$. The desired result immediately follows. □

B.2 Theorems

Theorem B.1 (Restatement of Theorem 3.1)**.** *If $0 < p_i \leq 1/5, \forall i \in [K]$, then the following holds regarding the expected class radius of each image in the latent space of our generative model:*

$$\mathbb{E}\left[r(z)\right] \leq \frac{\log\left(4\pi \log\left(1/\left\|p\right\|_2^2\right)\right)}{\sqrt{2\log\left(1/\left\|p\right\|_2^2\right)}}$$

Proof. If $p_i \leq 1/5$, then by Lemma B.2, we have the following:

$$\begin{aligned}
\Phi^{-1}\left(p_i\right) &\leq -\sqrt{\log\left(\frac{1/p_i^2}{4\pi\log\left(1/p_i\right)}\right)} \\
-\Phi^{-1}\left(p_i\right)^2 &\leq -\log\left(\frac{1/p_i^2}{4\pi\log\left(1/p_i\right)}\right) \\
e^{-\Phi^{-1}(p_i)^2} &\leq \frac{4\pi\log\left(1/p_i\right)}{1/p_i^2} \\
e^{-\Phi^{-1}(p_i)^2/2} &\leq \sqrt{\frac{4\pi\log\left(1/p_i\right)}{1/p_i^2}} \\
&= p_i\sqrt{4\pi\log\left(1/p_i\right)}
\end{aligned}$$

We therefore have:

$$\begin{aligned}
f(p) &\leq \sum_{i=1}^{K} -p_i\sqrt{\log\left(\frac{1/p_i^2}{4\pi\log\left(1/p_i\right)}\right)} + p_i\sqrt{2\log\left(1/p_i\right)} \\
&= \sum_{i=1}^{K} p_i\left(\sqrt{2\log\left(1/p_i\right)} - \sqrt{\log\left(\frac{1/p_i^2}{4\pi\log\left(1/p_i\right)}\right)}\right) \\
&= \sum_{i=1}^{K} p_i\left(\sqrt{2\log\left(1/p_i\right)} - \sqrt{2\log\left(1/p_i\right) - \log\left(4\pi\log\left(1/p_i\right)\right)}\right) \\
&= \sum_{i=1}^{K} p_i\left(\frac{\log\left(4\pi\log\left(1/p_i\right)\right)}{\sqrt{2\log\left(1/p_i\right)} + \sqrt{2\log\left(1/p_i\right) - \log\left(4\pi\log\left(1/p_i\right)\right)}}\right) \\
&\leq \sum_{i=1}^{K} p_i\left(\frac{\log\left(4\pi\log\left(1/p_i\right)\right)}{\sqrt{2\log\left(1/p_i\right)}}\right)
\end{aligned}$$

By Lemma B.6, using weights p_i, we can apply Jensen's Inequality:

$$\begin{aligned}
f(p) &\leq g\left(\left\|p\right\|_2^2\right) \\
&= \frac{\log\left(4\pi\log\left(1/\left\|p\right\|_2^2\right)\right)}{\sqrt{2\log\left(1/\left\|p\right\|_2^2\right)}}
\end{aligned}$$

as desired. □

Theorem B.2 (Restatement of Theorem 3.2)**.** *The upper bound stated in Theorem 3.1 is minimized when $p_i = 1/K$. Informally, if the generative model induces the uniform distribution over possible images, then the expected class radius is smallest.*

Proof. By Lemma B.7, if we wish to minimize g, it suffices to minimize $\left\|p\right\|_2^2$ as long as $\left\|p\right\|_2^2 < 1/2$. By Lemma B.5, this must be satisfied. Now, observe that for a vector $p \in \mathbb{R}^K$,

we must have $\|p\|_1 \leq \sqrt{K}\|p\|_2$. Since $\|p\|_1 = 1$, we have $1/\|p\|_2^2 \geq K$. We then use this in the bound from Theorem 3.1 to obtain the desired result.

We also present an information-theoretic proof. It is well-known that Renyi entropies achieve their maximum of $\log(K)$ when the distribution p is uniform (intuitively, this means that the distribution maximizing uncertainty is the uniform distribution). We then use the fact that the following function is clearly decreasing:

$$\frac{\log(4\pi x)}{\sqrt{2x}}$$

and we arrive at the same conclusion.

To obtain the exact form of the bound, use the fact that setting $p_i = 1/K$ yields $\|p\|_2^2 = 1/K$. □

Theorem B.3 (Restatement of Theorem 3.3)**.** *Suppose that we impose the following constraint on p_i:*

$$p_i \in \left[0, \min\left(\frac{1}{5}, h(K)\right)\right]$$

where $h(K)$ is a function of K such that $h(K) = o(1)$ and $h(K) = \Omega(1/K)$. Then, we have:

$$\lim_{K\to\infty} \mathbb{E}[r(z)] = 0$$

Proof. By Theorem 3.1, we have:

$$\mathbb{E}[r(z)] \leq \frac{\log\left(4\pi \log\left(1/\|p\|_2^2\right)\right)}{\sqrt{2\log\left(1/\|p\|_2^2\right)}}$$

Then, using the results from Lemmas B.5 and B.7, we have:

$$\mathbb{E}[r(z)] \leq \frac{\log\left(4\pi \log\left(\min\left(\frac{1}{5}, 3h(K)\right)\right)\right)}{\sqrt{2\log\left(1/\min\left(\frac{1}{5}, 3h(K)\right)\right)}}$$

Using Lemma B.7 along with the fact that $h(K) = o(1)$ yields the following:

$$\lim_{K\to\infty} \frac{\log\left(4\pi \log\left(\min\left(\frac{1}{5}, 3h(K)\right)\right)\right)}{\sqrt{2\log\left(1/\min\left(\frac{1}{5}, 3h(K)\right)\right)}} = 0$$

Since $\mathbb{E}[r(z)] \geq 0$, the desired result must follow by the squeeze theorem. □