ON THE NEED FOR TOPOLOGY-AWARE GENERATIVE MODELS FOR MANIFOLD-BASED DEFENSES

Anonymous authors
Paper under double-blind review

ABSTRACT

ML algorithms or models, especially deep neural networks (DNNs), have shown significant promise in several areas. However, recently researchers have demonstrated that ML algorithms, especially DNNs, are vulnerable to adversarial examples (slightly perturbed samples that cause mis-classification). Existence of adversarial examples has hindered deployment of ML algorithms in safety-critical sectors, such as security. Several defenses for adversarial examples exist in the literature. One of the important classes of defenses are manifold-based defenses, where a sample is "pulled back" into the data manifold before classifying. These defenses rely on the manifold assumption (data lie in a manifold of lower dimension than the input space). These defenses use a generative model to approximate the input distribution. This paper asks the following question: do the generative models used in manifold-based defenses need to be topology-aware? Our paper suggests the answer is yes. We provide theoretical and empirical evidence to support our claim.

1 INTRODUCTION

Machine learning (ML) algorithms, especially deep-neural networks (DNNs), have had resounding success in several domains. However, adversarial examples have hindered their deployment in safety-critical domains, such as autonomous driving and malware detection. Adversarial examples are constructed by an adversary an adding a small perturbation to a data-point so that it is misclassified. Several algorithms for constructing adversarial examples exist in the literature (Biggio et al., 2013; Szegedy et al., 2013; Goodfellow et al., 2014b; Kurakin et al., 2016a; Carlini & Wagner, 2017; Madry et al., 2017; Papernot et al., 2017). Numerous defenses for adversarial examples also have been explored (Kurakin et al., 2016b; Guo et al., 2017; Sinha et al., 2017; Song et al., 2017; Tramèr et al., 2017; Xie et al., 2017; Dhillon et al., 2018; Raghunathan et al., 2018; Cohen et al., 2019; Dubey et al., 2019).

In this paper we focus on "manifold-based" defenses (Ilyas et al., 2017; Samangouei et al., 2018). The general idea in these defenses is to "pull back" the data point into the data manifold before classification. These defenses leverage the fact that in several domains, natural data lies in a low-dimensional manifold (henceforth referred to as the manifold assumptions) (Zhu & Goldberg, 2009). The data distribution and hence actual manifold that the natural data lies in is usually unknown, so these defenses use a generative model to "approximate" the data distribution. Generative models attempt to learn to generate data according to the underlying data distribution. (The input to a generative model is usually random noise from a known distribution, such as Gaussian or uniform.) There are various types of generative models in the literature, such as VAE (Kingma & Welling, 2013), GAN (Goodfellow et al., 2014a) and reversible generative models, e.g. Real NVP (Dinh et al., 2016).

This paper addresses the following question:

> Do manifold-based defenses need to be aware of the topology of the underlying data manifold?

This paper suggests that the answer to this question is **yes**. We demonstrate that if the generative model does not capture the topology of the underlying manifold, it can adversely affect these defenses. In these cases, the underlying generative model is being used as an approximation of the underlying manifold. We believe that this opens a rich avenue for future work on using topology-aware generative models for defense to adversarial examples.

Contributions and Roadmap. We begin with a brief description of related work in Section 2. Section 3 provides the requisite mathematical background. Our main theoretical results are provided in Section 4. Informally, our result says that if the generative model is not topology-aware, it can lead to "topological mismatch" between the distribution induced by the generative model and the actual distribution. Section 5 experimentally verifies our theoretical results, and investigates their ramifications on a manifold-based defenses called Invert-and-Classify (INC) (Ilyas et al., 2017; Samangouei et al., 2018).

2 RELATED WORK

2.1 GENERATIVE MODELS

As a method for sampling high dimensional data, generative models find applications in various fields in applied math and engineering, e.g. image processing, reinforcement learning, etc. Learning data-generating distribution with neural networks includes well-known examples of *Variational Autoencoder (VAE)* (Kingma & Welling, 2013) and variations of *Generative Adversarial Network (GAN)* (Goodfellow et al., 2014a; Radford et al., 2015; Zhao et al., 2016).

These generative models learn how to map latent variables into generated samples. VAE is a variational Bayesian approach, so it approximates a posterior distribution over latent vectors (given training samples) by a simpler variational distribution. As other variational Bayesian methods do, VAE tries to minimize the KL divergence between the posterior distribution and the variational distribution by minimizing the reconstruction error of the auto-encoder. GAN is another type of approach to learning how to transform latent vectors into samples. Unlike other approaches, GAN learns the target distribution by training two networks – generator and discriminator – simultaneously.

In addition to generating plausible samples, some generative models construct bijective relation between latent vector and generated samples, so that the probability density of the generated sample can be estimated. Due to its bijective nature, such generative models are *reversible*. Some examples are *normalizing flow* (Rezende & Mohamed, 2015), *Masked Autoregressive Flow (MAF)* (Papamakarios et al., 2017), *Real NVP* (Dinh et al., 2016), and *Glow* (Kingma & Dhariwal, 2018).

2.2 APPLICATIONS OF GENERATIVE MODELS IN ADVERSARIAL MACHINE LEARNING

The DNN based classifier has been shown to be vulnerable to adversarial attacks (Szegedy et al., 2013; Goodfellow et al., 2014b; Moosavi-Dezfooli et al., 2016; Papernot et al., 2016; Madry et al., 2017). Several hypothesis try explaining such vulnerability (Szegedy et al., 2013; Goodfellow et al., 2014b; Tanay & Griffin, 2016; Feinman et al., 2017), and one explanation is that the adversarial examples lie far away from the data manifold. This idea leads to defense approaches making use of the geometry learned from the dataset – by projecting the given input to the nearest point in the data manifold.

To learn a manifold from a given dataset, generative models can be exploited. The main idea is to approximate the data-generating distribution with a generative model, to facilitate searching over data manifold by searching over the space of latent vectors. The term *invert and classify (INC)* was coined to describe this type of defense (Ilyas et al., 2017), and different types of generative models were tried to detect adversarial examples (Ilyas et al., 2017; Song et al., 2017; Samangouei et al., 2018). Usually, the projection is done by searching the latent vector that minimizes the geometric distance (Ilyas et al., 2017; Samangouei et al., 2018). However, despite the promising theoretical background, it turned out that all of those methods are still vulnerable (Athalye et al., 2018; Ilyas et al., 2017).

3 BACKGROUND

We formally describe data generation, based on the well-known manifold assumption; data lies close to a manifold whose intrinsic dimension is much lower than that of the ambient space. In our model of data generation, we provide a formal definition of *data-generating manifold* M on which the data-generating distribution lies so that M conforms to the manifold assumption.

3.1 REQUIREMENTS

Real-world data tends to be noisy, so they do not easily correspond to an underlying manifold. We first focus on an ideal case where data is generated solely from the manifold M without noise.

In the setting of classification with l labels, we consider manifolds $M_1, \ldots, M_l \subset \mathbb{R}^n$ that correspond to the generation of data in each class $i \in \{1, \ldots, l\}$, respectively. We assume that those manifolds are pair-wise disjoint, i.e. $M_i \cap M_j = \varnothing$ for any $i \neq j$.We set the data-generating manifold M as the disjoint union of those manifolds, $M = \biguplus_{i=1}^{l} M_i$. We assume M to be a compact Riemannian manifold with a volume measure dM induced by its Riemannian metric. When a density function p_M defined on M satisfies some requirements, it is possible to compute probabilities over M via $\int_{\mathbf{x} \in M} p_M(\mathbf{x}) dM(\mathbf{x})$. We call such M equipped with p_M an dM as a *data-generating manifold*. We refer to Appendix A and Appendix D.1 for details about definitions and requirements on p_M.

In practice, data generation is affected by noise, so not all data lie on the data-generating manifold. Therefore, we incorporate the noise as an artifact of data-generation and extend the density p_M on M to the density p on the entire $\mathbb{R}^n$ by assigning local noise densities on M. We consider a procedure that (1) samples a point $\mathbf{x}_o$ from M first, (2) adds a noise vector $\mathbf{n}$ to get an observed point $\hat{\mathbf{x}} = \mathbf{x}_o + \mathbf{n}$. Here, the noise $\mathbf{n}$ is a random vector sampled from a probability distribution, centered at $\mathbf{x}_o$, whose *noise density function* is $\nu_{\mathbf{x}_o}$, satisfying $\nu_{\mathbf{x}}(\mathbf{n}) = \nu_{\mathbf{x}}(\hat{\mathbf{x}} - \mathbf{x}) = p_M(\hat{\mathbf{x}} | \mathbf{x}_o = \mathbf{x})$.

3.2 Extending density

When M is equipped with a density function p_M and measure dM that we can integrate over M, we can compute the density after random noise is added as follows.

$$p(\hat{\mathbf{x}}) = \int_{\mathbf{x} \in M} \nu_{\mathbf{x}}(\hat{\mathbf{x}} - \mathbf{x}) p(\mathbf{x}) dM(\mathbf{x}) \tag{1}$$

Since $\nu_{\mathbf{x}}(\hat{\mathbf{x}} - \mathbf{x})$ is a function on $\mathbf{x}$ when $\hat{\mathbf{x}}$ is fixed, computing this integration can be viewed as computing expectation of a real-valued function defined on M. Computing such expectation has been explored in Pennec (1999). A demonstrative example is provided in Appendix B, and this extension is further discussed in Appendix D.2.

3.3 Generative models

A generative model tries to find a statistical model for joint density $p(\mathbf{x}, y)$ (Ng & Jordan, 2002). We mainly discuss a specific type that learns a transform from one distribution $\mathcal{D}_Z$ to another target distribution $\mathcal{D}_X$. Commonly, a latent vector $\mathbf{z} \sim \mathcal{D}_Z$ is sampled from a simpler distribution, e.g. Gaussian, then a pre-trained deterministic function G maps to a sample $\mathbf{x} = G(\mathbf{z})$.

Specifically, we focus on *reversible generative models* to facilitate the comparison between the density of generated samples and the target density. In this approach, the dimensions of latent vectors is set to be the same as that of the samples to be generated. Also, for a given $\mathbf{x}$, the density of $\mathbf{x}$ is estimated by the *change of variable formula* (equation (2) in Section 5.1).

3.4 Invert and classify (INC) approach for robust classification

As the data-generating manifold M contains class-wise disjoint manifolds, there is a classifier f on $\mathbb{R}^n$ separating these manifolds. If f separates the manifolds of M, any misclassified point should lie out of M. Therefore, to change a correct classification near a manifold, any adversary would pull a sample further out of the manifold. By projecting misclassified points to the nearest manifold, we may expect the classification to be corrected by the projection. *Invert and Classify (INC)* method (Ilyas et al., 2017; Samangouei et al., 2018) implements this using a generative model.

The main idea of INC is to invert the perturbed sample by projecting to the nearest point on data-generating manifold. Ideally, the data-generating manifold M is accessible. For any point $(\hat{\mathbf{x}}, y)$ with $f(\hat{\mathbf{x}}) \neq y$, out-of-manifold perturbation is reduced by projecting $\hat{\mathbf{x}}$ to $\mathbf{x}^*$ on M. The manifold M is unknown in practice. However, as M is the data-generating manifold of $\mathcal{D}_X$, a generative model G for $\mathcal{D}_X$ is trained to approximate M. Then, searching over M is replaced by searching over latent vectors of G. More details about INC implementations are described in Section 5.1.

4 Topological properties of data from generative models

In this paper, we study the significance of differences in the topological properties of the latent vector distribution and the target distribution, in learning generative models. Initial information about the topology of target distribution[1], is crucial to the generative model performance. Specifically, if there is a difference in the number of connected components in the superlevel set between the target

[1]The term *topology of distributions*, refers to the topology of shapes that correspond to the distributions.

distribution and the distribution of the latent vector, then any continuous generative model G cannot approximate the target distribution properly (irrespective of the training method). Due to the space limit, all proofs are presented in Appendix C.

4.1 Topology of distributions based on superlevel sets

The data-generating manifold is a geometric shape that corresponds to the distribution. However, this manifold is not accessible in most cases and we only have indirect access via the distribution extended from it. Therefore, we consider finding a shape from the extended density, so that this "shape" successfully approximates the data-generating manifold.

λ-density superlevel set. We use the concept of λ-density superlevel set to capture geometric features of the density function. Simply put, for a density function p and a threshold $\lambda > 0$, *λ-density superlevel set* $L_{p,\lambda}$ is the inverse image $p^{-1}[\lambda, \infty]$. Our theoretical contribution is the conditional existence of a λ-density superlevel set reflecting the topology of the data-generating manifold, under proper conditions on the noise density.

Assumptions on noise density. For a family of densities $\{\nu_{\mathbf{x}}\}_{\mathbf{x} \in M}$, we require the noise $\nu_{\mathbf{x}}$ to satisfy a number of assumptions. These assumptions facilitate theoretical discussion about the superlevel set reflecting the data-generating manifold. In the following definition, we denote a Euclidean ball of radius δ centered at $\mathbf{x}$ by $B_\delta(\mathbf{x})$.

Definition 1. Let $\nu_{\mathbf{x}}$ be a family of noise densities.

- λ is *small-enough* if $L_{\nu_{\mathbf{x}},\lambda}$ is nonempty for all $\mathbf{x} \in M$,
- *λ-bounding radius* $\delta_{\mathbf{x},\lambda} := \min\{\delta \mid L_{\nu_{\mathbf{x}},\lambda} \subseteq \overline{B_\delta(\mathbf{0})}\}$ is the smallest radius that $\overline{B_\delta(\mathbf{0})}$ contains $L_{\nu_{\mathbf{x}},\lambda}$. When $\max_{\mathbf{x} \in M} \delta_{\mathbf{x},\lambda}$ exists for some λ, we denote the maximum value as δ_λ.
- *λ-guaranteeing radius* $\epsilon_{\mathbf{x},\lambda} := \max\{\epsilon \mid \overline{B_\epsilon(\mathbf{0})} \subseteq L_{\nu_{\mathbf{x}},\lambda}\}$ is the largest radius that $L_{\nu_{\mathbf{x}},\lambda}$ contains $\overline{B_\epsilon(\mathbf{0})}$. When $\min_{\mathbf{x} \in M} \epsilon_{\mathbf{x},\lambda}$ exists for some λ, we denote the minimum value as ϵ_λ.

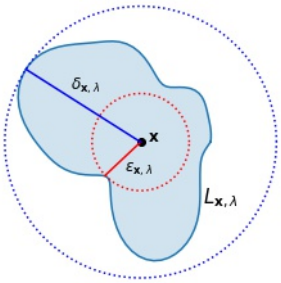


Figure 1: Example superlevel set $L_{\mathbf{x},\lambda}$ with λ-bounding radius $\delta_{\mathbf{x},\lambda}$ and λ-guaranteeing radius $\epsilon_{\mathbf{x},\lambda}$.

Sufficient conditions for the existence of these radii are discussed in Appendix D.3. The properties of these radii are summarized in Lemma 1. (The proof follows from Definition 1).

Lemma 1. Let $\nu_{\mathbf{x}}$ be a family of noise densities and let λ be small-enough. Then,

$$\|\hat{\mathbf{x}} - \mathbf{x}\| > \delta_\lambda \implies \nu_{\mathbf{x}}(\hat{\mathbf{x}} - \mathbf{x}) < \lambda$$
$$\|\hat{\mathbf{x}} - \mathbf{x}\| \leq \epsilon_\lambda \implies \nu_{\mathbf{x}}(\hat{\mathbf{x}} - \mathbf{x}) \geq \lambda$$

whenever δ_λ and ϵ_λ exist.

Figure 1 shows an example of superlevel set $L_{\mathbf{x},\lambda}$ of noise $\nu_{\mathbf{x}}$ at a point $\mathbf{x}$ and its λ-bounding radius $\delta_{\mathbf{x},\lambda}$ and λ-guaranteeing radius $\epsilon_{\mathbf{x},\lambda}$.

Finally, we define the continuous variation of noise densities $\nu_{\mathbf{x}}$ over changes of $\mathbf{x} \in M$. For the continuous variation, we require the continuity of both radii $\delta_{\mathbf{x},\lambda}$ and $\epsilon_{\mathbf{x},\lambda}$, as real-valued functions of $\mathbf{x} \in M$ for any fixed value of λ.

Definition 2 (Continuously varying radii)**.** Noise densities $\nu_{\mathbf{x}}$ have *continuously varying radii* if, for a fixed small-enough λ, both λ-bounding radius $\delta_{\mathbf{x},\lambda}$ and λ-guaranteeing radius $\epsilon_{\mathbf{x},\lambda}$ are continuous functions of $\mathbf{x} \in M$.

When noise densities have continuously varying radii, with the compactness of M, we can apply the extreme value theorem to guarantee the existence of both $\delta_\lambda = \max_{\mathbf{x} \in M} \delta_{\mathbf{x},\lambda}$ and $\epsilon_\lambda = \min_{\mathbf{x} \in M} \epsilon_{\mathbf{x},\lambda}$.

4.2 Main theorem

Our main theorem establishes, under the assumptions on noise densities from Section 4.1, the existence of a λ such that,

- **(Inclusion)** The λ-density superlevel set $L_{p,\lambda}$ includes the data-generating manifold M.
- **(Separation)** The λ-density superlevel set $L_{p,\lambda}$ consists of connected components such that each component contains at most one manifold M_i.

Definition 3. Consider a data-generating manifold M with density function p_M. For a radius $\epsilon > 0$, we define ω_ϵ to be the minimum (over $\mathbf{x} \in M$) probability of sampling $\mathbf{x}' \in M$ in an ϵ-ball $B_\epsilon(\mathbf{x})$.

$$\omega_\epsilon := \min_{\mathbf{x}\in M} \Pr_{\mathbf{x}'\sim p_M} [\mathbf{x}' \in B_\epsilon(\mathbf{x})]$$

Definition 4 (Class-wise distance)**.** Let (X, d) be a metric space and let $M = \dot{\bigcup}_{i=1}^{l} M_i$ be a data-generating manifold in X. The class-wise distance d_{cw} of M is defined as,

$$d_{\text{cw}} = \min_{\substack{i,j\in[l] \\ i\neq j}} \min_{\substack{\mathbf{x}\in M_i \\ \mathbf{x}'\in M_j}} d(\mathbf{x}, \mathbf{x}')$$

With the definitions above, we proved the following main theorem.

Theorem 1. Pick any small-enough threshold λ. Fix a value $\lambda^* \leq \omega_\epsilon \lambda$ and let $\delta^* = \delta_{\lambda^*}$ be the λ^*-bounding radius. If d_{cw} of M is larger than $2\delta^*$, then the superlevel set L_{p,λ^*} satisfies the following properties.

- L_{p,λ^*} contains the data-generating manifold M.
- Each connected component of L_{p,λ^*} contains at most one manifold M_i of class i.

4.3 Application to the generative model

We show an application of Theorem 1. We denote the target distribution by $\mathcal{D}_X$, the latent distribution by $\mathcal{D}_Z$, and the distribution of $G(\mathbf{z})$ where $\mathbf{z} \sim \mathcal{D}_Z$ by $\mathcal{D}_{G(Z)}$. Similarly, we denote the corresponding λ-density superlevel sets of densities by L_λ^X, L_λ^Z, and $L_\lambda^{G(Z)}$. We assume the generative model G to be continuous. Then, we get the following theorem regarding the difference between L_λ^X and $L_\lambda^{G(Z)}$, in the number of connected components. [2]

Theorem 2. Let $\mathcal{D}_Z$ be a mixture of n_Z multivariate Gaussian distributions, and let the data-generating manifold of $\mathcal{D}_X$ contain n_X components. Let G be a continuous generative model for $\mathcal{D}_X$ using latent vectors from $\mathcal{D}_Z$. Let λ^* be the threshold value from the Theorem 1. If $n_Z < n_X$, $L_{\lambda^*}^X$ and $L_{\lambda^*}^{G(Z)}$ do not agree on the number of connected components.

We can use this theorem to deduce the need for proper information about the target distribution when training a generative model, especially if it is used for a security-critical application, e.g. INC.

Corollary 1. If Theorem 2 is satisfied, there is a point $\hat{\mathbf{x}} \in \mathbb{R}^n$ such that $\hat{\mathbf{x}} \notin L_{\lambda^*}^X$ but $\hat{\mathbf{x}} \in L_{\lambda^*}^{G(Z)}$.

As a result, with density at least λ^*, G generates a point $\hat{\mathbf{x}}$ which is unlikely to be generated by the target distribution. Since INC is based on generations of G, INC method can output an out-of-manifold point as a solution of optimization (12).

5 Experimental results

In this section, we empirically demonstrate the consequence of the two theorems and explore their implication for the INC defense. Our main goals are to provide (1) empirical support for the applicability of Theorem 2 and Corollary 1 via toy datasets, and (2) the improvement in INC performance using a class-aware generative model. The main questions and the corresponding answers are,

(Q1) Can we experimentally verify the results of section 4.3? Specifically, can we find cases that the superlevel sets of $\mathcal{D}_X$ and $\mathcal{D}_{G(Z)}$ have different numbers of connected components?

(Q2) How does INC fail when the generative model is ignorant of topology information?

(Q3) Does the class-aware generative model improve the INC performance?

(A1) Theorem 2 and Corollary 1 can be verified by plotting λ-density superlevel set. Especially, we visualize the λ-density superlevel set of $\mathcal{D}_{G(Z)}$ reflecting Theorem 2 and Corollary 1.

[2] In Appendix D.4, Theorem 2 is generalized for more topological properties.

(**A2**) When generative model is not trained with topology information, naive INC may fail. We found out two possible reasons regarding INC failure: (1) choice of a bad initial point and (2) out-of-manifold search due to non-separation of density superlevel set.

(**A3**) The performance of INC is improved by training generative models with topology information on the target distribution. We improved the average INC performance by decreasing the error induced by projection to 30% compared to the class-ignorant counterpart.

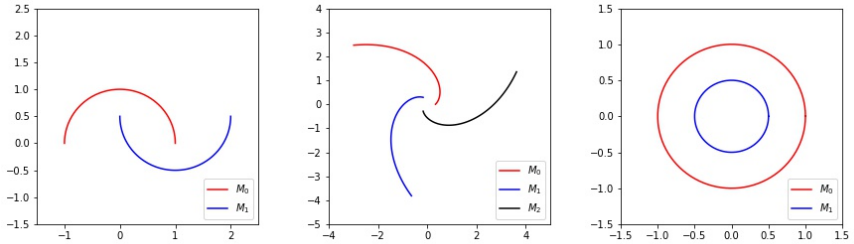
In the rest of this section, we provide a more detailed description of our experiments. First, we briefly describe the experimental setup in section 5.1: datasets, latent vector distributions, training method, and INC implemetation. Then, section 5.2-5.4 describe the experimental results regarding the findings summarized above. Section 5.5 contains an additional experiment illustrating the changes of decision boundaries by INC application.

5.1 Experimental setup

Datasets. For all experiments, we use three toy datasets in $\mathbb{R}^2$: two-moons, spirals, and circles. Table 1 summarizes the parameterizations[3] of each data-generating manifold and Figure 2 are the plots of the corresponding data-generating manifolds. To construct the training set, we first sampled 1000

two-moons	spirals	circles
$M_0 : \left\{ (x_1, x_2) \middle\vert \begin{array}{l} x_1 = \cos\theta \\ x_2 = \sin\theta \end{array} \right\}$	$M_0 : \left\{ (x_1, x_2) \middle\vert \begin{array}{l} x_1 = \frac{1}{3} e^t \cos(t) \\ x_2 = \frac{1}{3} e^t \cos(t) \end{array} \right\}$	$M_0 : \left\{ (x_1, x_2) \middle\vert \begin{array}{l} x_1 = \cos\theta \\ x_2 = \sin\theta \end{array} \right\}$
$M_1 : \left\{ (x_1, x_2) \middle\vert \begin{array}{l} x_1 = 1 - \cos\theta \\ x_2 = 1 - \sin\theta + \frac{1}{2} \end{array} \right\}$	$M_1 : \left\{ (x_1, x_2) \middle\vert \begin{array}{l} x_1 = \frac{1}{3} e^t \cos(t + \frac{2}{3}\pi) \\ x_2 = \frac{1}{3} e^t \sin(t + \frac{2}{3}\pi) \end{array} \right\}$	$M_1 : \left\{ (x_1, x_2) \middle\vert \begin{array}{l} x_1 = \frac{1}{2} \cos\theta \\ x_2 = \frac{1}{2} \sin\theta \end{array} \right\}$
for $\theta \in [0, \pi]$	$M_2 : \left\{ (x_1, x_2) \middle\vert \begin{array}{l} x_1 = \frac{1}{3} e^t \cos(t + \frac{4}{3}\pi) \\ x_2 = \frac{1}{3} e^t \sin(t + \frac{4}{3}\pi) \end{array} \right\}$	for $\theta \in [0, 2\pi]$
	for $t \in [0, T]$ where $T = \ln\left(\frac{15}{\sqrt{2}} + 1\right)$	

Table 1: Parameterizations of dataset used in the experiments.



(a) two-moons (b) spirals (c) circles

Figure 2: Data-generating manifolds used in the experiments

points uniformly from each manifold M_i, then each point was perturbed by isotropic Gaussian noise $\mathcal{N}(\mathbf{0}, \sigma^2 I_2)$ with $\sigma = 0.05$. Before the training, each training set was standardized by a preprocessing of Scikit-learn package.

Latent vector distributions. For latent vector distributions $\mathcal{D}_Z$, we prepared three different mixtures of n_Z Gaussian distributions with $n_Z \in \{1, 2, 3\}$. When $n_Z = 1$, we simply use $\mathcal{N}(\mathbf{0}, I_2)$. When $n_Z = 2, 3$, we arranged n_Z Gaussian distributions along a circle of radius $R = 2.5$, so that i-th Gaussian has mean at $\mu_i = \left(-R \sin\left(\frac{2\pi i}{n}\right), R\cos\left(\frac{2\pi i}{n}\right)\right)$ with $\sigma = 0.5$ for $n = 2$ and $\sigma = 0.3$ for $n = 3$. Then, the uniform mixtures of the arranged Gaussian were used as $\mathcal{D}_Z$. In Figure 3 (top row), we visualize the connected components corresponding to the latent vector distributions.

Training generative models. Our experiments mostly use the Tensorflow Probability (Dillon et al., 2017) library which contains the implementation of reversible generative models. Specifically,

[3]The value T is from the reparameterization $t = \ln\left(s/\sqrt{2} + 1\right)$ for $s \in [0, 15]$ for uniform sampling.

the Tensorflow Probability library contains an implementation of the Real NVP coupling layer that we used as the building blocks of our models. The default template provided by Tensorflow Probability library was used to construct each Real NVP coupling layer with two hidden layers of 128 units. Each model uses eight coupling layers that are followed by permutation exchanging two dimensions of $\mathbb{R}^2$ except for the last coupling layer.

We describe the details of the training procedure of the generative models used in the experiments. We prepared two different types of generative models: class-ignorant and class-aware.

The *class-ignorant* type is the usual Real NVP model. This model uses the empirical estimation of log-likelihood as over a training batch $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ as its training loss.

$$\ell_{\text{ci}} = \frac{1}{m} \sum_{t=1}^{m} \log(p_X(\mathbf{x}_t))$$

where p_X is estimated by applying *the change of variables formula*,

$$p_X(\mathbf{x}) = p_Z(\mathbf{z}) \left| \det \left(\frac{\partial G(\mathbf{z})}{\partial \mathbf{z}^T} \right) \right|^{-1} \tag{2}$$

where p_X, p_Z are the probability densities (of $\mathcal{D}_X, \mathcal{D}_Z$, respectively), and $\frac{\partial G(\mathbf{z})}{\partial \mathbf{z}^T}$ is the Jacobian of G as a function from $\mathbb{R}^n$ to itself.

The *class-aware* type is the Real NVP model trained with information about the number of connected components, i.e. the number of class labels l Using the number of labels, the densities p_X and p_Z can be decomposed as follows.

$$\begin{aligned} p_X(\mathbf{x}) &= \sum_{i \in \{1,\ldots,l\}} \Pr[y = i]\, p_{X,i}(\mathbf{x}) \\ p_Z(\mathbf{z}) &= \sum_{i \in \{1,\ldots,l\}} \Pr[y = i]\, p_{Z,i}(\mathbf{z}) \end{aligned} \tag{3}$$

where $p_{X,i}(\mathbf{x}) = p_X(\mathbf{x}|y = i)$ and each $p_{Z,i}$ is the i-th Gaussian component described above. Since $\Pr[y = i]$ is not generally known, the uniform distribution $\Pr[y = i] = \frac{1}{l}$ is used, where l is the number of classification labels.

The main idea is class-wise training, i.e. training each $p_{X,i}$ from each $p_{Z,i}$. Applying the change of variables formula for each class i,

$$p_{X,i}(\mathbf{x}) = p_{Z,i}(\mathbf{z}) \left| \det \left(\frac{\partial G(\mathbf{z})}{\partial \mathbf{z}^T} \right) \right|^{-1} \tag{4}$$

Combining equations (3) and (4), we get the change of variables formula (2), so this approach works as a reversible generative model allowing the density computation.

We define the class-wise loss function ℓ_i for class-wise training as follows.

$$\ell_i = \frac{1}{m_i} \sum_{t=1}^{m} \mathbb{1}[y_t = i] \ \log(p_{X,i}(\mathbf{x}_t))$$

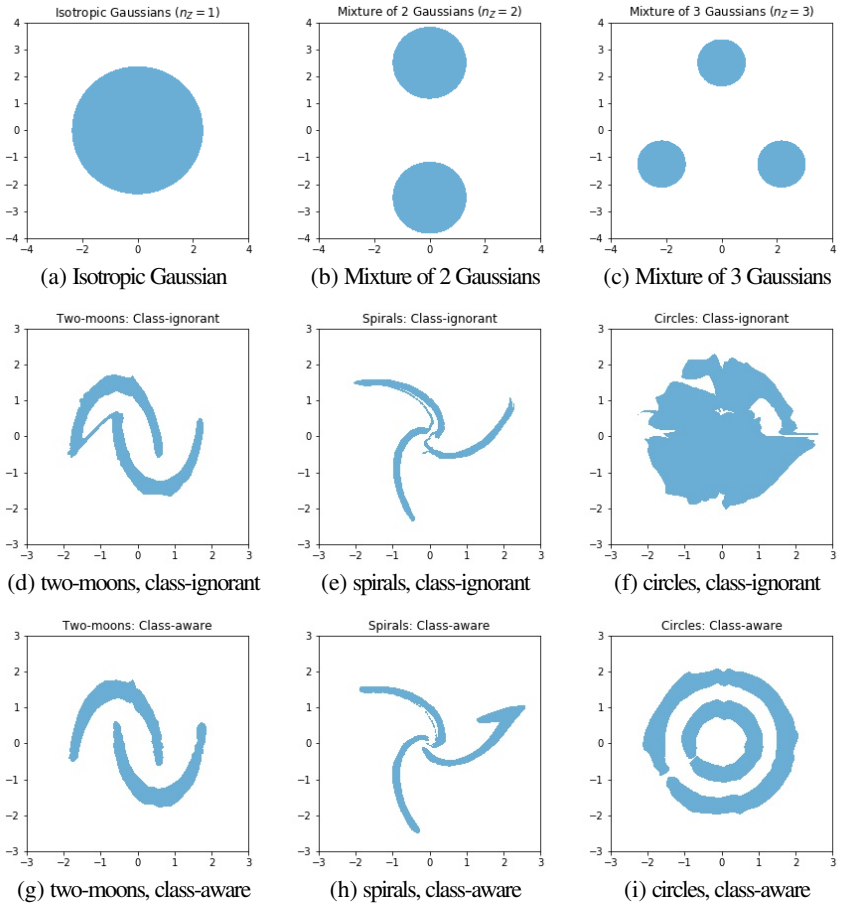
where m_i is the number of training samples in class i. Then, we train a generative model using the weighted sum of ℓ_i as the training loss function.

$$\ell_{\text{ca}} = \sum_{i \in \{1,\ldots,l\}} \Pr[y = i]\, \ell_i$$

Each model was trained for 30000 iterations. For each iteration, a batch of 200 random samples was chosen from two-moons and circles dataset, and a batch of 300 random samples was chosen from the spirals dataset. For the choices of latent vector distribution, we chose the mixture of $l - 1$ Gaussians for class-ignorant type, whereas we chose the mixture of l Gaussians for class-aware type.

5.2 Visual verification of theorems

The goal of this section is to verify the Theorem 2 and the Corollary 1, by visualizing the superlevel set reflecting the statements. Figure 3 shows the λ-density superlevel sets of densities of $\mathcal{D}_{G(Z)}$ using the same threshold $\lambda = 0.01$. The first row and the second row show the results from the



(a) Isotropic Gaussian (b) Mixture of 2 Gaussians (c) Mixture of 3 Gaussians

(d) two-moons, class-ignorant (e) spirals, class-ignorant (f) circles, class-ignorant

(g) two-moons, class-aware (h) spirals, class-aware (i) circles, class-aware

Figure 3: λ-density superlevel sets of $\mathcal{D}_Z$ and $\mathcal{D}_{G(Z)}$ with $\lambda = 0.01$. Top row: $\mathcal{D}_Z$ for $n_Z = 1, 2, 3$. Middle row: $\mathcal{D}_{G(Z)}$, class-ignorant model. Bottom row: $\mathcal{D}_{G(Z)}$, class-aware model.

class-ignorant version and those from the class-aware version, respectively. Each column corresponds to each dataset. All distributions are scaled for the standardization preprocessing before the training.

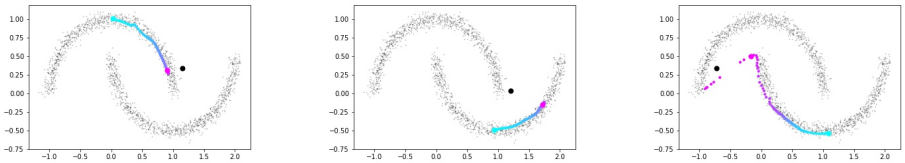
In general, superlevel set components are separated when the generative model is class-aware. On the contrary, the class-ignorant generative models introduce connections between superlevel set components, as anticipated by the Corollary 1. Due to this connection, the class-ignorant generative models achieve less number of connected components in its superlevel set, and this verifies the Theorem 2 for our choice of $\lambda^* = 0.01$.

5.3 INC FAILURE DUE TO THE LACK OF INFORMATION ON THE DISTRIBUTION TOPOLOGY

We present how the non-separation of superlevel set components influences the performance of the INC. We provide two possible explanations of why INC fails. First is the bad initialization causing a suboptimal solution on a manifold not-the-nearest to the input. The second is due to an artifact, induced by the topological difference producing an out-of-manifold solution.

Figure 4 presents three visualized examples of INC with a class-ignorant generative model for two-moons. In each plot, the black dot is the given point $\hat{\mathbf{x}}$, and cyan dot is the initial point from choosing $\mathbf{z}$ randomly from the latent vector distribution – $\mathcal{N}(\mathbf{0}, I_2)$, and magenta dot is the final point outputted by INC. All intermediate points of the optimization are plotted with dots, changing colors gradually from cyan to magenta. The training set for two-moon used in the training procedure is plotted in gray.

Figure 4a is the INC optimization with an ideal start. The initial point lies in the same manifold as the manifold closest to $\hat{\mathbf{x}}$. Then, the INC optimization searches along the manifold, converging to a point close to $\hat{\mathbf{x}}$. Figure 4b is the case that INC fails because of a bad initialization. The initial point was chosen on the other manifold than the manifold containing the desired solution, so INC



(a) INC with an ideal initialization (b) INC with a bad initialization (c) INC searching out of manifold

Figure 4: Successful and failed cases of INC using class-ignorant generative model of two-moon.

converged to a local optimum on the wrong manifold. Our class-aware INC performs manifold-wise initialization to circumvent this issue. Figure 4c is the case that INC failed due to out-of-manifold search. Not only that INC converged in a wrong manifold, but a nontrivial amount of intermediate points also lie out of manifold, resulting in an out-of-manifold solution (see Figure 3d).

5.4 INC IMPROVEMENT VIA CLASS-AWARE GENERATIVE MODEL

We demonstrate that INC performance is improved by using class-aware generative models. To measure the performance of INC, 100 points are chosen uniformly from each manifold M_i. Then, each point $\mathbf{x}$ is perturbed by $\mathbf{n}_{\mathbf{x}}$ normal to the manifold at $\mathbf{x}$, generating 200 adversarial points $\hat{\mathbf{x}} = \mathbf{x} \pm r\,\mathbf{n}_{\mathbf{x}}$. For all datasets, $r = 0.2$ is used for perturbation size. We expect two types of INC to map $\hat{\mathbf{x}}$ back to the original point $\mathbf{x}$, as $\mathbf{x}$ is the optimal solution to (11). We define the *projection error* of INC as $\|\text{INC}(\hat{\mathbf{x}}) - \mathbf{x}\|_2$, and collect the statistics of projection errors over all $\hat{\mathbf{x}}$.

Two-moons		Spirals		Circles	
class-ignorant	class-aware	class-ignorant	class-aware	class-ignorant	class-aware
0.647 (0.666)	0.148 (0.208)	1.523 (1.338)	0.443 (0.440)	0.699 (0.491)	0.180 (0.259)

Table 2: Comparison of the projection errors of INC based on the class-awareness of the model.

Table 2 shows the projection error statistics for two types of generative model. Each pair of columns is the results on the indicated dataset. For each pair, one column shows the performance of the class-ignorant INC and the other column shows that of the class-aware counterpart. Numbers in each cell are average and standard deviation (in parenthesis) of the projection error. For any dataset, class-aware INC achieves significantly lower projection errors. Histograms of the projection errors are provided in Appendix E.

5.5 ADDITIONAL EXPERIMENTS FOR THE INC PERFORMANCE.

Finally, we present experiments to demonstrate the effect of the superlevel set discrepancy on the INC performance. First, we begin with training support vector machines (SVMs) performing classification tasks for our target distributions. For training data, we randomly sampled 1000 training points from each data-generating manifold. The baseline SVMs were intentionally ill-trained, by using high kernel coefficient $\gamma = 100$. [4] After training SVMs, we form other classifiers by applying INC to ill-trained SVMs To explain, for each dataset, we have four types of classifiers as follows.

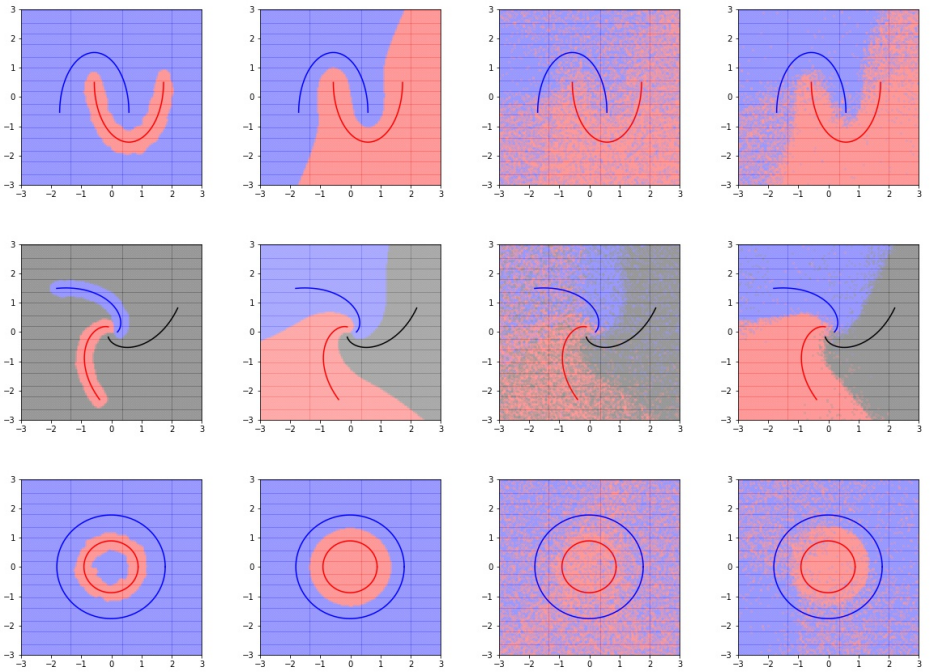
- **(1)** Ill-trained SVM: Baseline classifier
- **(2)** Ideal INC: Classifier with INC using a direct access to the data-generating manifolds
- **(3)** topology-ignorant INC: Classifier with INC using a topology-ignorant generative model
- **(4)** topology-aware INC: Classifier with INC with using a topology-aware generative model

We want to emphasize that direct access to the data-generating manifold is not possible in general. However, applying INC based on such direct access gives us an INC purely based on the geometry, so it is an ideal form of INC that should be approximated. Also, since topology-ignorant is affected by a bad choice of an initial point, we reduce the effect of bad initialization by sampling more initial

[4] In general, choosing unnecessarily high kernel coefficient γ causes overfitting (Chaudhuri et al., 2017), inducing decision boundary close to the training data.

points and taking the best solution among the projection results. For this number of initial choices, we chose initial points as many as the number of manifolds, which is exactly the same as the number of initial points for the topology-aware INC model.

To demonstrate the improvement in the robustness of the model, we visualize the effect by depicting the decision boundary of each classifier. To be specific, we form a 300×300 grid on the domain of $[-3, 3] \times [-3, 3]$, then compute the result of classification. The depicted decision boundaries are presented in Figure 5. Each row corresponds to each dataset: two moons, spirals, and circles, respectively. Each column corresponds to classifier 1-4 described above, from the first column to the fourth column, respectively. From Figure 5, it is visually evident that topology-aware INC models provide more proper approximations to the ideal INC model.



(a) Ill-trained SVM (b) Ideal INC (c) topology-ignorant INC (d) topology-aware INC

Figure 5: Changes in the decision boundaries of ill-trained SVM after INC applications.

6 Conclusion

We theoretically and experimentally discussed the necessity of topology-awareness in the training of generative models, especially in security-critical applications. A continuous generative model is sensitive to the topological mismatch between the latent vector distribution and the target distribution. Such mismatch leads to potential problems with manifold-based adversarial defenses utilizing generative models such as INC. We described two cases when INC fails: the bad initialization and the artifacts from the topological difference. We experimentally verified that topology-aware training effectively prevents these problems, thereby improving the effectiveness of generative models in manifold-based defense. After topology-aware training of generative models, the INC projection errors fell down to 30% of the errors of the topology-ignorant INC.

References

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European*

conference on machine learning and knowledge discovery in databases, pp. 387–402. Springer, 2013.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57. IEEE, 2017.

Arin Chaudhuri, Deovrat Kakde, Carol Sadek, Laura Gonzalez, and Seunghyun Kong. The mean and median criteria for kernel bandwidth selection for support vector data description. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 842–849. IEEE, 2017.

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems*, pp. 6571–6583, 2018.

Jeremy M Cohen, Elan Rosenfeld, and J Zico Kolter. Certified adversarial robustness via randomized smoothing. *arXiv preprint arXiv:1902.02918*, 2019.

Guneet S Dhillon, Kamyar Azizzadenesheli, Zachary C Lipton, Jeremy Bernstein, Jean Kossaifi, Aran Khanna, and Anima Anandkumar. Stochastic activation pruning for robust adversarial defense. *arXiv preprint arXiv:1803.01442*, 2018.

Joshua V Dillon, Ian Langmore, Dustin Tran, Eugene Brevdo, Srinivas Vasudevan, Dave Moore, Brian Patton, Alex Alemi, Matt Hoffman, and Rif A Saurous. Tensorflow distributions. *arXiv preprint arXiv:1711.10604*, 2017.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Abhimanyu Dubey, Laurens van der Maaten, Zeki Yalniz, Yixuan Li, and Dhruv Mahajan. Defense against adversarial images using web-scale nearest-neighbor search. *arXiv preprint arXiv:1903.01612*, 2019.

Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017.

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, 2014a.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014b.

Will Grathwohl, Ricky TQ Chen, Jesse Betterncourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens Van Der Maaten. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117*, 2017.

Andrew Ilyas, Ajil Jalal, Eirini Asteri, Constantinos Daskalakis, and Alexandros G Dimakis. The robust manifold defense: Adversarial training using generative models. *arXiv preprint arXiv:1712.09196*, 2017.

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.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pp. 10215–10224, 2018.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016a.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016b.

J.M. Lee. *Introduction to Smooth Manifolds*. Graduate Texts in Mathematics. Springer, 2003. ISBN 9780387954486. URL `https://books.google.com/books?id=eqfgZtjQceYC`.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2574–2582, 2016.

J.R. Munkres. *Topology*. Featured Titles for Topology Series. Prentice Hall, Incorporated, 2000. ISBN 9780131816299. URL `https://books.google.com/books?id=XjoZAQAAIAAJ`.

Andrew Y Ng and Michael I Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*, pp. 841–848, 2002.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pp. 2338–2347, 2017.

Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 372–387. IEEE, 2016.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pp. 506–519. ACM, 2017.

Xavier Pennec. Probabilities and statistics on riemannian manifolds: Basic tools for geometric measurements. In *NSIP*, pp. 194–198. Citeseer, 1999.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. *arXiv preprint arXiv:1801.09344*, 2018.

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. *arXiv preprint arXiv:1805.06605*, 2018.

Aman Sinha, Hongseok Namkoong, and John Duchi. Certifiable distributional robustness with principled adversarial training. *stat*, 1050:29, 2017.

Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766*, 2017.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Thomas Tanay and Lewis Griffin. A boundary tilting persepective on the phenomenon of adversarial examples. *arXiv preprint arXiv:1608.07690*, 2016.

Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

E Weinan. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5(1):1–11, 2017.

Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. *arXiv preprint arXiv:1711.01991*, 2017.

Linfeng Zhang, Lei Wang, et al. Monge-amp\ere flow for generative modeling. *arXiv preprint arXiv:1809.10188*, 2018.

Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016.

Xiaojin Zhu and Andrew B Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009.

A MATHEMATICAL BACKGROUND

A.1 GENERAL TOPOLOGY

We introduce definitions and theorems related to general topology appeared in the paper. For more details, all the definitions and theorems can be found in Munkres (2000).

Definitions in general topology. We first provide the precise definitions of the terms we brought from the general topology.

Definition 5 (Topological space)**.** A *topology* on a set X is a collection $\mathcal{T}$ of subsets of X having the following properties.

1. $\varnothing$ and X are in $\mathcal{T}$.
2. The union of the elements of any subcollection of $\mathcal{T}$ is in $\mathcal{T}$.
3. The intersection of the elements of any finite subcollection of $\mathcal{T}$ is in $\mathcal{T}$.

A set X for which a topology $\mathcal{T}$ has been specified is called a *topological space*.

For example, a collection of all open sets in $\mathbb{R}^n$ is a topology, thus $\mathbb{R}^n$ is a topological space. If a topology can be constructed by taking arbitrary union and a finite number of intersections of a smaller collection $\mathcal{B}$ of subsets of X, we call $\mathcal{B}$ is a basis of the topology.

Pick a metric d in $\mathbb{R}^n$ and consider $\mathcal{B}$ a set of all open balls in $\mathbb{R}^n$ using the metric d. The topology of $\mathbb{R}^n$ can be constructed by taking $\mathcal{B}$ as a basis. When this construction is possible, metric d is said to *induce the topology*.

Definition 6 (Metrizable space)**.** If X is a topological space, X is said to be *metrizable* if there exists a metric d on the set X that induces the topology of X. A *metric space* is a metrizable space X together with a specific metric d that gives the topology of X.

Since $\mathbb{R}^n$ is equipped with Euclidean metric that induces its topology, $\mathbb{R}^n$ is metrizable.

Continuity and the extreme value theorem. Let X and Y be topological spaces In the field of general topology, a function $f : X \to Y$ is said to be *continuous*, if for any subset V open in Y, its inverse image $f^{-1}(V)$ is open in X. Moreover, if f is a continuous bijection whose inverse is also continuous, f is called a *homeomorphism*. The notion of homeomorphism is important as it always preserves topological property, e.g. connectedness, compactness, etc., and this will be used in the further generalization of Theorem 2.

Here, we only introduce the generalized statement of extreme value theorem.

Theorem 3 (Extreme value theorem)**.** Let $f : X \to Y$ be continuous, where Y is an ordered set. If X is compact, then there exist points $\underline{\mathbf{x}}$ and $\overline{\mathbf{x}}$ in X such that $f(\underline{\mathbf{x}}) \leq f(\mathbf{x}) \leq f(\overline{\mathbf{x}})$ for every $\mathbf{x} \in X$.

Specifically, if a manifold M is a compact subset in $\mathbb{R}^n$, we may use $X = M$ and $Y = \mathbb{R}$.

Normal space and Urysohn lemma. The Urysohn lemma was used to prove the Corollary 1. We first introduce the notion of normal space.

Definition 7 (Normal space)**.** Let X be a topological space that one-point sets in X are closed. Then, X is *normal* if for each pair A, B of disjoint closed sets of X, there exist disjoint open sets containing A and B, respectively.

Urysohn's lemma is another equivalent condition for a space to be normal.

Theorem 4 (Urysohn lemma)**.** Let X be a normal topological space; let A and B be disjoint closed subsets in X. Let $[a, b]$ be a closed interval in the real line. Then there exists a continuous map

$$f : X \longrightarrow [a, b]$$

such that $f(\mathbf{x}) = a$ for every $\mathbf{x}$ in A, and $f(\mathbf{x}) = b$ for every $\mathbf{x}$ in B.

To apply this lemma to $\mathbb{R}^n$, we only need the following theorem.

Theorem 5. Every metrizable space is normal.

Since $\mathbb{R}^n$ is metrizable, it is a normal space by Theorem 5. Therefore, we can apply Urysohn lemma to any pair of disjoint subsets in $\mathbb{R}^n$, to show the existence of a continuous map $f : X \to [0, 1]$.

A.2 DIFFERENTIAL GEOMETRY

We provide the definitions from differential geometry (Lee, 2003) used in the paper.

Manifold and tangent space.

Definition 8 (Manifold)**.** Suppose M is a topological space. We say M is a topological manifold of dimension k if it has the following properties.

1. For any pair of distinct points $\mathbf{x}_1, \mathbf{x}_2 \in M$, there are disjoint open subsets $U_1, U_2 \subset M$ such that $\mathbf{x}_1 \in U$ and $\mathbf{x}_2 \in V$.
2. There exists a countable basis for the topology of M.
3. Every point has a neighborhood U that is homeomorphic to an open subset $\tilde{U}$ of $\mathbb{R}^k$.

There are different ways to define tangent space of k-dimensional manifold M. Informally, it can be understood as *geometric tangent space* to $M \subset \mathbb{R}^n$ at a point $\mathbf{x} \in M$, which is a collection of pairs $(\mathbf{x}, \mathbf{v})$ where $\mathbf{v}$ is a vector tangentially passing through $\mathbf{x}$. Here we put a more formal definition of tangent space. Consider a vector space $C^\infty(M)$, a set of smooth functions on M.

Definition 9 (Tangent space)**.** Let $\mathbf{x}$ be a point of a smooth manifold M. A linear map $X : C^\infty(M) \to \mathbb{R}$ is called a *derivation at* $\mathbf{x}$ if it satisfies

$$X(fg) = f(\mathbf{x})Xg + g(\mathbf{x})Xf$$

for all $f, g \in C^\infty(M)$.

The set of all derivations of $C^\infty(M)$ at $\mathbf{x}$ forms a vector space called the *tangent space* to M at $\mathbf{x}$, and is denoted by $T_\mathbf{x}(M)$.

Riemannian metric. As tangent space $T_\mathbf{x}(M)$ is a vector space for each $\mathbf{x} \in M$, we can consider a inner product g_{bfx} defined on $T_\mathbf{x}(M)$.

Definition 10 (Riemannian metric)**.** A *Riemannian metric* g on a smooth manifold M is a smooth collection of inner products $g_\mathbf{x}$ defined for each $T_\mathbf{x}(M)$. The condition for smoothness of g is that, for any smooth vector fields $\mathcal{X}, \mathcal{Y}$ on M, the mapping $\mathbf{x} \mapsto g_\mathbf{x}(\mathcal{X}|_\mathbf{x}, \mathcal{Y}|_\mathbf{x})$.

A manifold M equipped with a Riemannian metric g is called a Riemannian manifold.

B EXAMPLES

Computing density p_M over a Riemannian manifold M. This section presents example computations of the probability computations from the Section D.1 and 3.2 As a concrete example of computing density over a manifold, we use the following simple manifolds, so called *two-moons* in $\mathbb{R}^2$.

$$M_0 = \left\{(x_1, x_2) \middle| \begin{array}{l} x_1 = \cos\theta \\ x_2 = \sin\theta \end{array} \text{ for } \theta \in [0, \pi]\right\}$$
$$M_1 = \left\{(x_1, x_2) \middle| \begin{array}{l} x_1 = 1 - \cos\theta \\ x_2 = 1 - \sin\theta + \frac{1}{2} \end{array} \text{ for } \theta \in [0, \pi]\right\}$$

We take $M = M_0 \cup M_1$ as our example manifold. Figure 6a shows the manifold of two-moons dataset plotted in different colors: M_0 in red and M_1 in blue.

First recall the following equation (equation (8) from the Section D.1).

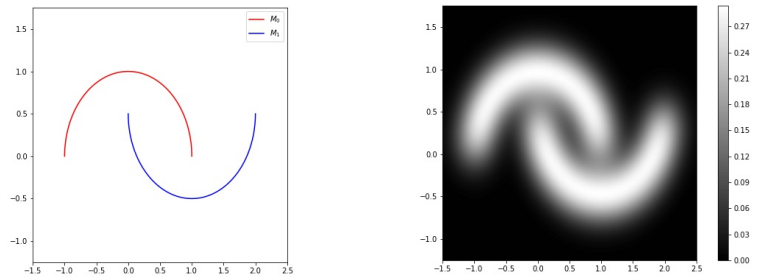
$$\int_{\mathbf{x} \in M} p_M(\mathbf{x}) dM(\mathbf{x}) = \int_{\mathbf{u} \in D} p_M(X(\mathbf{u})) \sqrt{|\det[g_{X(\mathbf{u})}]|} d\mathbf{u}$$

where $[g_{X(\mathbf{u})}]$ is the $k \times k$ matrix representation of the inner product $g_{X(\mathbf{u})}$ at $X(\mathbf{u}) \in M$.

Especially, when a manifold in $\mathbb{R}^n$ is of dimension 1, i.e. parameterized curve $\gamma : [a, b] \to \mathbb{R}^n$, the integration (8) can be written in simpler way.

$$\int_{\mathbf{x} \in M} p_M(\mathbf{x}) dM(\mathbf{x}) = \int_{t=a}^{b} p_M(\gamma(t)) \|\gamma'(t)\| dt \tag{5}$$

where $\gamma'(t)$ is the n-dimensional velocity vector at $t \in [a, b]$.



(a) Plot of the two-moons manifold in $\mathbb{R}^2$

(b) Extended density function over $\mathbb{R}^2$ from the two-moons dataset

Figure 6: Density extension example from two-moons manifold.

Let p_M be a probability density function defined on M. As M is composed of two disjoint manifolds M_0 and M_1, we consider conditional densities p_0, p_1 as follows.

$$\begin{aligned} p_0(\mathbf{x}) &= p_M(\mathbf{x} \mid \mathbf{x} \in M_0) = \frac{p_M|_{M_0}(\mathbf{x})}{\Pr[\mathbf{x} \in M_0]} \\ p_1(\mathbf{x}) &= p_M(\mathbf{x} \mid \mathbf{x} \in M_1) = \frac{p_M|_{M_1}(\mathbf{x})}{\Pr[\mathbf{x} \in M_1]} \end{aligned} \tag{6}$$

Here, $p_M|_{M_0}$ and $p_M|_{M_1}$ are the density function p_M with its domain restricted to M_0 and M_1, respectively. By our definition of data-generating manifolds, $\Pr[\mathbf{x} \in M_i]$ corresponds to the probability of data generation for class i, i.e. $\Pr[y = i]$. For a concrete example of such density, uniform density for each manifold M_i can be defined as $p_i(\mathbf{x}) = \frac{1}{\pi}$ for all $\mathbf{x} \in M_i$.

Note that each of manifolds are parameterized curves in $\mathbb{R}^2$,

$$\begin{aligned} \gamma_0 &: \theta \mapsto (\cos\theta, \sin\theta) \\ \gamma_1 &: \theta \mapsto (1 - \cos\theta, 1 - \sin\theta + 0.5) \end{aligned}$$

with constant speed $\|\gamma_0'(\theta)\| = \|\gamma_1'(\theta)\| = 1$ at all $\theta \in [0, \pi]$. Therefore, from equation (5),

$$\begin{aligned} \int_{\mathbf{x} \in M_0} p_M|_{M_0}(\mathbf{x}) dM_0(\mathbf{x}) &= \int_{\theta=0}^{\pi} p_M(\gamma_0(\theta)) d\theta \\ \int_{\mathbf{x} \in M_0} p_M|_{M_1}(\mathbf{x}) dM_1(\mathbf{x}) &= \int_{\theta=0}^{\pi} p_M(\gamma_1(\theta)) d\theta \end{aligned} \tag{7}$$

For any measurable subset $A \subseteq M$, the probability for an event that $\mathbf{x}$ is in A can be computed as follows.

$$\begin{aligned} \Pr[\mathbf{x} \in A] &= \int_{\mathbf{x} \in A \subseteq M} p_M(\mathbf{x}) dM(\mathbf{x}) \\ &= \int_{\mathbf{x} \in A \cap M_0} p_M|_{M_0}(\mathbf{x}) dM_0(\mathbf{x}) + \int_{\mathbf{x} \in A \cap M_1} p_M|_{M_1}(\mathbf{x}) dM_1(\mathbf{x}) \\ &= \int_{\substack{\theta \in [0,\pi] \\ \gamma_0(\theta) \in A}} p_M|_{M_0}(\gamma_0(\theta)) d\theta + \int_{\substack{\theta \in [0,\pi] \\ \gamma_1(\theta) \in A}} p_M|_{M_1}(\gamma_1(\theta)) d\theta \quad (\because (7)) \\ &= \Pr[\mathbf{x} \in M_0] \int_{\substack{\theta \in [0,\pi] \\ \gamma_0(\theta) \in A}} p_0(\gamma_0(\theta)) d\theta \\ &\quad + \Pr[\mathbf{x} \in M_1] \int_{\substack{\theta \in [0,\pi] \\ \gamma_1(\theta) \in A}} p_1(\gamma_1(\theta)) d\theta \quad (\because (6)) \\ &= \frac{1}{\pi} \left(\Pr[\mathbf{x} \in M_0] \int_{\substack{\theta \in [0,\pi] \\ \gamma_0(\theta) \in A}} 1 d\theta + \Pr[\mathbf{x} \in M_1] \int_{\substack{\theta \in [0,\pi] \\ \gamma_1(\theta) \in A}} 1 d\theta \right) \end{aligned}$$

We can briefly check all the requirements (**R1**), (**R2**), and (**R3**). The computation of $\Pr[\mathbf{x} \in A]$ is based on (**R1**), so (**R1**) is satisfied trivially. Also, p_M is a function defined only on M, thus (**R2**) is clear, i.e. $\mathrm{supp}(p_M) = \{\mathbf{x} \in \mathbb{R}^n \mid p(\mathbf{x}) > 0\} \subseteq M$. To check (**R3**), when $A = M_i$, computing this integration will result in the exact probability $\Pr[\mathbf{x} \in M_i] = \Pr[y = i]$, so when $A = M$, computing the integration will result in $\Pr[y = 0] + \Pr[y = 1] = 1$, as desired in the requirements.

Extending density to $\mathbb{R}^n$. We extend the domain to $\mathbb{R}^n$ for the example of two-moon. In Section 3, we defined the noise density function to satisfy the following requirement.

(**R0**) The *translated noise density function*, $\nu_{\mathbf{x}}(\hat{\mathbf{x}} - \mathbf{x})$, is the density of noise $\mathbf{n} = \hat{\mathbf{x}} - \mathbf{x}$ being chosen for a given $\mathbf{x}$. Given $\mathbf{x}_o = \mathbf{x}$, since adding noise $\mathbf{n}$ is the only way to generate $\hat{\mathbf{x}}$ by perturbing $\mathbf{x}_0$, $p(\hat{\mathbf{x}}|\,\mathbf{x}_o = \mathbf{x})$ is equal to $\nu_{\mathbf{x}}(\mathbf{n})$.

We show an example that under a proper noise density function, we can construct the density extended from M satisfying the requirement (**R0**). For simplicity, we choose isotropic Gaussian distribution, $\mathcal{N}(0, \sigma^2 I)$ with standard deviation σ for each dimension, as the noise density function $\nu_{\mathbf{x}}$ for all $\mathbf{x} \in M$. Such noise density $\nu_{\mathbf{x}}$ defined in $\mathbb{R}^n$ can be written as follows.

$$\nu_{\mathbf{x}}(\mathbf{n}_{\mathbf{x}}) = \frac{1}{\sqrt{2\pi}\sigma^2} \exp\left(-\frac{\|\,\mathbf{n}_{\mathbf{x}}\,\|_2^2}{2\sigma^2}\right)$$

By putting $\mathbf{n}_{\mathbf{x}} = \hat{\mathbf{x}} - \mathbf{x}$ to density equation above,

$$p(\hat{\mathbf{x}}) = \int_{\mathbf{x}\in M} \frac{1}{\sqrt{2\pi}\sigma^2} \exp\left(-\frac{\|\hat{\mathbf{x}} - \mathbf{x}\,\|_2^2}{2\sigma^2}\right) p_M(\mathbf{x}) dM(\mathbf{x})$$

Specifically, We assume an isotropic Gaussian distribution with $\sigma = 0.05$ as the noise density $\nu_{\mathbf{x}}$ for all $\mathbf{x} \in M$.

By the equation (1), we have the following computation of density on $\hat{\mathbf{x}}$.

$$\begin{aligned}
p(\hat{\mathbf{x}}) &= \int_{\mathbf{x}\in M} \nu_{\mathbf{x}}(\hat{\mathbf{x}} - \mathbf{x}) p_M(\mathbf{x}) dM(\mathbf{x}) \\
&= \int_{\mathbf{x}\in M_0} \nu_{\mathbf{x}}(\hat{\mathbf{x}} - \mathbf{x}) p_M|_{M_0}(\mathbf{x}) dM_0(\mathbf{x}) + \int_{\mathbf{x}\in M_1} \nu_{\mathbf{x}}(\hat{\mathbf{x}} - \mathbf{x}) p_M|_{M_1}(\mathbf{x}) dM_1(\mathbf{x}) \\
&= \int_{\theta=0}^{\pi} \nu_{\mathbf{x}}(\hat{\mathbf{x}} - \mathbf{x}) p_M|_{M_0}(\gamma_0(\theta)) d\theta + \int_{\theta=0}^{\pi} \nu_{\mathbf{x}}(\hat{\mathbf{x}} - \mathbf{x}) p_M|_{M_1}(\gamma_1(\theta)) d\theta \quad (\because (5)) \\
&= \Pr[\mathbf{x} \in M_0] \int_{\theta=0}^{\pi} \nu_{\mathbf{x}}(\hat{\mathbf{x}} - \mathbf{x}) p_0(\gamma_0(\theta)) d\theta \\
&\quad + \Pr[\mathbf{x} \in M_1] \int_{\theta=0}^{\pi} \nu_{\mathbf{x}}(\hat{\mathbf{x}} - \mathbf{x}) p_1(\gamma_1(\theta)) d\theta \quad (\because (6)) \\
&= \frac{1}{\pi\sqrt{2\pi}\sigma^2} \Bigg[\Pr[\mathbf{x} \in M_0] \int_{\theta=0}^{\pi} \exp\left(-\frac{\|\hat{\mathbf{x}} - \mathbf{x}\,\|_2^2}{2\sigma^2}\right) d\theta \\
&\quad + \Pr[\mathbf{x} \in M_1] \int_{\theta=0}^{\pi} \exp\left(-\frac{\|\hat{\mathbf{x}} - \mathbf{x}\,\|_2^2}{2\sigma^2}\right) d\theta \Bigg]
\end{aligned}$$

We can also check that the requirement (**R0**) is satisfied by the construction; our construction (equation (1)) is based on (**R0**). The computed density is shown in Figure 6b.

C PROOFS

In this Section, we provide the proofs for statements that appeared in Section 4.

C.1 PROOF OF THEOREM 1

To begin with, pick a value λ such that the λ-density superlevel set $L_{\nu_{\mathbf{x}},\lambda}$ is nonempty for all $\mathbf{x} \in M$. As we use noise densities $\nu_{\mathbf{x}}$ described in Section 4.1, it is safe to assume that both λ-bounding radius $\delta_\lambda = \max_{\mathbf{x}\in M} \delta_{\mathbf{x},\lambda}$ and λ-guaranteeing radius $\epsilon_\lambda = \min_{\mathbf{x}\in M} \epsilon_{\mathbf{x},\lambda}$ exist.

Then, we can prove that, with a proper choice of threshold λ, λ-density superlevel set includes the data-generating manifold.

Lemma 2. Assume that noise densities have radii in Definition 1 for all $\mathbf{x} \in M$ and a small enough $\lambda > 0$. Then, for any $\mathbf{x} \in M$, the density $p(\mathbf{x})$ is at least $\omega_\epsilon \lambda$, i.e. $p(\mathbf{x}) \geq \omega_\epsilon \lambda$, where $\epsilon = \epsilon_\lambda$.

Proof. By Lemma 1,

$$\begin{aligned}
\mathbf{x}' \in B_\epsilon(\mathbf{x}) &\iff \mathbf{x} \in B_\epsilon(\mathbf{x}') = B_{\epsilon_\lambda}(\mathbf{x}') \quad (\because \epsilon = \epsilon_\lambda) \\
&\implies \nu_{\mathbf{x}'}(\mathbf{x} - \mathbf{x}') \geq \lambda
\end{aligned}$$

Then, we can lower bound the density $p_M(\mathbf{x})$ as follows.

$$\begin{aligned} p(\mathbf{x}) &= \int_{\mathbf{x}' \in M} \nu_{\mathbf{x}'}(\mathbf{x}-\mathbf{x}') p_M(\mathbf{x}') dM(\mathbf{x}') \\ &\geq \int_{\mathbf{x}' \in M \cap B_\epsilon(\mathbf{x})} \nu_{\mathbf{x}'}(\mathbf{x}-\mathbf{x}') p_M(\mathbf{x}') dM(\mathbf{x}') \\ &\geq \lambda \int_{\mathbf{x}' \in M \cap B_\epsilon} p_M(\mathbf{x}') dM(\mathbf{x}') \\ &= \lambda \Pr_{\mathbf{x}' \in M}[\mathbf{x}' \in B_\epsilon(\mathbf{x})] \\ &\geq \omega_\epsilon \lambda \end{aligned}$$

□

This lemma shows that the thresholding the extended density p with threshold $\lambda^* \leq \omega_\epsilon \lambda$ guarantees the superlevel set to include the entire manifold M.

Corollary 2. For any threshold $\lambda^* \leq \omega_\epsilon \lambda$, the corresponding λ^*-density superlevel set L_{p,λ^*} of the extended density p includes the data-generating manifold M.

Similarly, we show that, with a proper choice of threshold λ, each connected component of λ-density superlevel set contains at most one manifold.

Lemma 3. Assume a family of noise densities satisfies the assumptions of Section 4.1. Let $\lambda > 0$ be a value such that the λ-density superlevel set $L_{\nu_{\mathbf{x}},\lambda}$ is nonempty for any $\mathbf{x} \in M$. Also, let $\delta = \delta_\lambda$ be the maximum λ-bounding radius over M. Then, for any $\hat{\mathbf{x}} \notin N_\delta(M)$, the extended density value is smaller than λ, i.e. $p(\hat{\mathbf{x}}) < \lambda$.

Proof. By Lemma 1,

$$\begin{aligned} \hat{\mathbf{x}} \notin N_\delta(M) &\iff \hat{\mathbf{x}} \notin B_\delta(\mathbf{x}) = B_{\delta_\lambda}(\mathbf{x}) \text{ for any } \mathbf{x} \in M \quad (\because \delta = \delta_\lambda) \\ &\implies \nu_{\mathbf{x}}(\hat{\mathbf{x}}-\mathbf{x}) < \lambda \text{ for any } \mathbf{x} \in M \end{aligned}$$

Then, we can upper bound the density $p(\hat{\mathbf{x}})$ as follows.

$$\begin{aligned} p(\hat{\mathbf{x}}) &= \int_{\mathbf{x} \in M} \nu_{\mathbf{x}}(\hat{\mathbf{x}}-\mathbf{x}) p_M(\mathbf{x}) dM(\mathbf{x}) \\ &< \lambda \int_{\mathbf{x} \in M} p_M(\mathbf{x}) dM(\mathbf{x}) \quad (\because \hat{\mathbf{x}} \notin N_{\delta_\lambda}(M)) \\ &= \lambda \end{aligned}$$

□

This lemma says that, the λ-density superlevel set is included by the δ-neighborhood $N_\delta(M)$ of the data-generating manifold M.

Now, we can deduce the following main result.

Theorem 1. Pick any $\lambda^* \leq \omega_\epsilon \lambda$ threshold value satisfying the Corollary 2. If the class-wise distance of data-generating manifold is larger than $2\delta^*$ where $\delta^* = \delta_{\lambda^*}$(the λ^*-bounding radius), then the superlevel set L_{p,λ^*} satisfies the followings.

- L_{p,λ^*} contains the data-generating manifold M.
- Each connected component of L_{p,λ^*} contains at most one manifold M_i of class i.

Proof. The first property is a direct application of Corollary 2 for $\lambda^* = \omega_\epsilon \lambda$.

For the Second property, since the class-wise distance of M is larger than $2\delta^*$, the δ^*-neighborhood of manifolds are pairwise disjoint, i.e. $N_{\delta^*}(M_i) \cap N_{\delta^*}(M_j) = \varnothing$ for each $i \neq j$. Therefore, $N_{\delta^*}(M)$ has exactly k connected components $N_i = N_{\delta^*}(M_i)$'s.

By Lemma 3, δ^*-neighborhood $N_{\delta^*}(M)$ contains the superlevel set L_{p,λ^*}, thus each connected component of L_{p,λ^*} is in exactly one of N_i's. Since M is contained in L_{p,λ^*}, each M_i is contained in some connected component C of L_{p,λ^*} which is in N_i. Then, for any $j \neq i$, $M_j \notin C \subset N_i$, since M_j is in N_j which is disjoint to N_i. Therefore, if a connected component C contains a manifold M_i, then it cannot contain any other manifold. □

C.2 PROOFS FOR SECTION 4.3

Theorem 2. Let $\mathcal{D}_Z$ be a mixture of n_Z multivariate Gaussian distributions, and let $\mathcal{D}_X$ be the target distribution from a data-generating manifold with n_X manifolds. Let G be a continuous generative model for $\mathcal{D}_X$ using latent vectors from $\mathcal{D}_Z$. Assume the Theorem 1 is satisfied, and let λ^* be the threshold value from the Theorem 1. If $n_Z < n_X$, $L^X_{\lambda^*}$ and $L^{G(Z)}_{\lambda^*}$ do not agree on the number of connected components.

Proof. Since $L^X_{\lambda^*}$ is the results of Theorem 1, the number of connected components of $L^X_{\lambda^*}$ is at least n_X.

However, since $\mathcal{D}_Z$ is a mixture of Gaussians, for any value of λ (including the special case $\lambda = \lambda^*$), L^Z_λ can never have more than n_Z connected components. Since G is continuous, it preserves the number of connected components, thus $L^{G(Z)}_{\lambda^*} = G(L^Z_{\lambda^*})$ has at most n_Z connected components. As $n_Z < n_X$, $L^X_{\lambda^*}$ and $L^{G(Z)}_{\lambda^*}$ can never agree on the number of connected components. □

Corollary 1. If Theorem 2 is satisfied, there is a point $\hat{\mathbf{x}} \in \mathbb{R}^n$ such that $\hat{\mathbf{x}} \notin L^X_{\lambda^*}$ but $\hat{\mathbf{x}} \in L^{G(Z)}_{\lambda^*}$.

Proof. Since $n_Z < n_X$, there exists a connected components $\hat{C}$ of $L^{G(Z)}_{\lambda^*}$ containing at least two connected components of $S^X_{\lambda^*}$. Without loss of generality, assume $\hat{C}$ contains exactly two connected components C and C'. By definition, λ-superlevel set is a closed set, so C and C' are disjoint closed sets. In Euclidean space $\mathbb{R}^n$, the Urysohn lemma tells us that for any disjoint pair of closed sets A, A' in $\mathbb{R}^n$, there is a continuous function f such that $f|_A(\mathbf{x}) = 0$ and $f|_{A'}(\mathbf{x}) = 1$ for any $\mathbf{x} \in \mathbb{R}^n$. Especially, when $A = C$ and $A' = C'$, there exists a continuous function f such that,

- $f(\mathbf{x}) = 0$ for all $\mathbf{x}$ in C
- $f(\mathbf{x}) = 1$ for all $\mathbf{x}$ in C'

Consider $S = f^{-1}(\frac{1}{2})$ which is a separating plane separating C and C'. If $\hat{C} \cap S = \varnothing$, then $\hat{C} \cap S = f^{-1}[0, \frac{1}{2})$ and $\hat{C} \cap S = f^{-1}(\frac{1}{2}, 1]$ will be two open set in subspace $\hat{C}$, whose union is $\hat{C}$. This implies that $\hat{C}$ is disconnected, which is a contradiction. Therefore, $\hat{C} \cap S$ should be nonempty, and any point $\mathbf{x}$ in $\hat{C} \cap S$ is not in $L^X_{\lambda^*}$. □

D FURTHER DISCUSSIONS

D.1 COMPUTING DENSITY OVER A DATA-GENERATING MANIFOLD

When M is a Riemannian manifold equipped with a Riemannian metric g, we can compute probabilities over M. There are two essential components of probability computation: (a) a density function p_M and (b) a measure dM over M. We assume p_M and dM to satisfy the followings.

- **(R1)** For any measurable subset $A \subset M$, i.e. $\Pr[\mathbf{x} \in A] = \int_{\mathbf{x} \in A} p_M(\mathbf{x}) dM(\mathbf{x})$.
- **(R2)** p is zero everywhere out of M, i.e. $\mathrm{supp}(p_M) = \{\mathbf{x} \in \mathbb{R}^n \mid p_M(\mathbf{x}) > 0\} \subseteq M$
- **(R3)** For any $(\mathbf{x}, y)$, $\mathbf{x}$ is sampled from M_i if and only if $y = i$, i.e. $\Pr[\mathbf{x} \in M_i] = Pr[y = i]$

When equipped with such p_M and dM, we call M as a *data-generating manifold*.

Probability over a Riemannian manifold. We show how to compute a probability of $\mathbf{x}$ being generated from a Riemannian manifold M. We assume a k-dimensional manifold M equipped with a Riemannian metric g, a family of inner products $g_{\mathbf{x}}$ on tangent spaces $T_{\mathbf{x}}M$. In this case, g induces the volume measure dM for integration over M. If M is parameterized by $\mathbf{x} = X(\mathbf{u})$ for $\mathbf{u} \in D \subseteq \mathbb{R}^k$, the integration of a density function p_M on M is as follows.

$$\int_{\mathbf{x} \in M} p_M(\mathbf{x}) dM(\mathbf{x}) = \int_{\mathbf{u} \in D} p_M(X(\mathbf{u})) \sqrt{|\det[g_{X(\mathbf{u})}]|} d\mathbf{u} \tag{8}$$

where $[g_{X(\mathbf{u})}]$ is the $k \times k$ matrix representation of the inner product $g_{X(\mathbf{u})}$ at $X(\mathbf{u}) \in M$.

In section B, a concrete example of this computation will be provided.

D.2 DENSITY EXTENSION OF THE SECTION 3.2

This section introduces some remaining discussions regarding our data-generating process from a data-generating manifold.

Relation to kernel density estimation. While this extension is computing the density of compound distribution, it can be interpreted as computing expectation over a family of locally defined densities. Such expected value can be observed in previous approaches of density estimation. For example, if $\nu_{\mathbf{x}}$ is isotropic Gaussian for each $\mathbf{x}$, the above integration is equivalent to the kernel density estimation, with Gaussian kernel, over infinitely many points on M.

Observed property of the extended density. In Figure 6b in Section B, we can observe that the extended density achieved higher values near the data-generating manifold. We formalize this observation to discuss its implication to the INC approach.

Let $d(\hat{\mathbf{x}}, M)$ to be the minimum distance from $\hat{\mathbf{x}}$ to the manifold M.

(**C1**) For any given $\hat{\mathbf{x}}$, let y^* be the class label whose conditional density $p(\hat{\mathbf{x}}|y = y*)$ dominates $p(\hat{\mathbf{x}}|y = i)$ for $i \neq y^*$,

$$y^* \in \arg\max_{i \in [l]} p(\hat{\mathbf{x}}|y = i) \tag{9}$$

and let M_{y^*} be the manifold corresponding to y^*.

(**C2**) For y^* satisfying (**C1**), we choose y^* such that the distance of $\hat{\mathbf{x}}$ from the manifold $d(\hat{\mathbf{x}}, M_{y^*})$ is the smallest.

If there are multiple y^* satisfying both of (**C1**) and (**C2**), we expect the following property to be true for all of those y^*.

(**P1**) Consider the shortest line from $\hat{\mathbf{x}}$ to the manifold M_{y^*}. As $\hat{\mathbf{x}}$ goes closer to M_{y*} along this line, $\hat{\mathbf{x}}$ should be more likely to be generated as the influence of noise decreases when moving away from the manifold. Therefore, we expect our density p_M to have the following property.

$$\begin{aligned} &\mathbf{x}^* \in \arg\min_{\mathbf{x} \in M_{y^*}} d(\hat{\mathbf{x}}, \mathbf{x}) \\ \Longrightarrow\ &p(\hat{\mathbf{x}}) \leq p((1-\lambda)\hat{\mathbf{x}} + \lambda \mathbf{x}^*) \text{ for all } \lambda \in [0,1] \end{aligned} \tag{10}$$

Actually, this provides another justification of INC. In reality, the density conditioned by the label is not available even after running a generative model, so finding y^* with (**C1**) is relatively hard. If we only consider (**C2**) without filtering y^* via (**C1**), we are finding a point $\mathbf{x} \in M$ achieving the minimum distance to $\hat{\mathbf{x}}$, which is the optimization (11) above. Then projecting $\hat{\mathbf{x}}$ to the $\mathbf{x}^*$, i.e. the solution of the optimization 11, can be explained by 10; when $\lambda = 1$, p is the highest along the shortest line between $\hat{\mathbf{x}}$ and $\mathbf{x}^*$.

D.3 SUFFICIENT CONDITIONS FOR THE EXISTENCE OF RADII

We discuss the sufficient conditions guaranteeing the existence of radii introduced in Definition 1. Those sufficient conditions are derived from natural intuition about the properties of distributions in most machine learning context.

The first intuition is that the influence of noise should diminish as observed sample $\hat{\mathbf{x}}$ moves away from a source point $\mathbf{x}_o$. Therefore, we formalize the noise whose density decreases as the noise $\mathbf{n} = \hat{\mathbf{x}} - \mathbf{x}_o$ gets bigger. We formalize boundedness of noise densities via the boundedness of their λ-density superlevel sets and continuity of noise density via the continuity of individual $\nu_{\mathbf{x}}$.

Definition 11 (Center-peaked noise density)**.** Noise density functions $\nu_{\mathbf{x}}$ are *center-peaked*, if for any source point $\mathbf{x} \in M$ and any noise vector $\mathbf{n} \in \mathbb{R}^n$ with $\|\mathbf{n}\| > 0$, $\nu_{\mathbf{x}}(\mathbf{n}) < \nu_{\mathbf{x}}(\lambda \mathbf{n})$ for all $\lambda \in [0, 1)$

Definition 12 (Bounded noise density)**.** Noise density functions $\nu_{\mathbf{x}}$ are *bounded*, if a λ-density superlevel set is nonempty, there is a radius δ by which the λ-density superlevel set is bounded, i.e. $L_{\nu_{\mathbf{x}},\lambda} \subseteq \overline{B_\delta(\mathbf{0})}$ where $\overline{B_\delta(\mathbf{0})}$ is the closed ball of radius δ centered at $\mathbf{0}$.

Definition 13 (Continuous noise density)**.** Noise density functions $\nu_{\mathbf{x}}$ are *continuous*, if $\nu_{\mathbf{x}}$ is continuous in $\mathbb{R}^n$, for any $\mathbf{x} \in M$

Under the conditions above, the radii in Definition 1 always exist.

Proposition 1. If noise densities $\nu_{\mathbf{x}}$ are center-peaked, bounded, and continuous. Then, any nonempty λ-density superlevel set $L_{\nu_{\mathbf{x}},\lambda}$ has both λ-bounding radius $\delta_{\mathbf{x},\lambda}$ and λ-guaranteeing radius $\epsilon_{\mathbf{x},\lambda}$.

Proof. Let $\nu_{\mathbf{x}}$ be a center peaked, superlevel set bounded family of continuous noise densities. Since $\nu_{\mathbf{x}}$ is continuous, superlevel set $L_{\nu_{\mathbf{x}},\lambda} = \nu_{\mathbf{x}}^{-1}[\lambda, \infty)$ is closed as an inverse image of $\nu_{\mathbf{x}}$. Therefore, it's boundary $\partial L_{\nu_{\mathbf{x}},\lambda}$ is contained in $L_{\nu_{\mathbf{x}},\lambda}$.

Because $\nu_{\mathbf{x}}$ is superlevel set bounded, superlevel set $L_{\nu_{\mathbf{x}},\lambda}$ is bounded by a closed ball $\overline{B_\delta(\mathbf{0})}$ with radius $\delta \geq 0$. Since $\nu_{\mathbf{x}}$ is center peaked, a nonempty superlevel set $L_{\nu_{\mathbf{x}},\lambda}$ always contains $\mathbf{0}$ as the maximum is achieved at $\mathbf{0}$. Moreover, there exists a closed neighborhood ball $\overline{B_\epsilon(\mathbf{0})}$ with radius $\epsilon \geq 0$ contained in the superlevel set $L_{\nu_{\mathbf{x}},\lambda}$. Now it is enough to show that the minimum of δ and the maximum of ϵ exist.

Since $L_{\nu_{\mathbf{x}},\lambda}$ is bounded, its boundary $\partial L_{\nu_{\mathbf{x}},\lambda}$ is also bounded. $\partial L_{\nu_{\mathbf{x}},\lambda}$ is closed and bounded, thus it is a compact set. Therefore, the Euclidean norm, as a continuous function, should achieve the maximum $\overline{r}$ and the minimum $\underline{r}$ on $\partial L_{\nu_{\mathbf{x}},\lambda}$ by the extreme value theorem. From the choice of δ and ϵ, we can get,

$$\epsilon \leq \underline{r} \leq \overline{r} \leq \delta$$

Therefore, we can find the minimum $\delta_{\mathbf{x},\lambda} = \overline{r}$ and the maximum $\epsilon_{\mathbf{x},\lambda} = \underline{r}$. □

D.4 Generalization of the Theorem 1

We try generalizing the Theorem 2 to handle more concepts in topology. The Theorem 2 mainly uses a fact that the number of connected components of λ-density superlevel set is preserved by a continuous generative model G.

In algebraic topology, each connected component corresponds to a generator of 0-th homology group H_0, and continuity of a function is enough to preserve each component. In general, generators of i-th homology group H_i for $i > 0$ are not preserved by a continuous map, so we need to restrict G further. By requiring G to be a homeomorphism, we can safely guarantee that all topological properties are preserved by G, therefore we can generalize the Theorem 2 with a homeomorphic generative model G.

To generalize the proof of the Theorem 2, we first provide the sketch of the proof.

- **(1)** λ^*-density superlevel set $L^Z_{\lambda^*}$ of mixture of Gaussian has at most n_Z connected component.
- **(2)** Since G is continuous, the number of connected components of $L^G_{\lambda^*}(Z) = G(L^Z_{\lambda^*})$ is same to the number of connected components of $L^Z_{\lambda^*}$, so it is also at most n_Z.
- **(3)** We chose λ^* so that $L^X_{\lambda^*}$ is included in δ^*-neighborhood of M.
- **(4)** By assumption on the class-wise distance of M, δ^*-neighborhood of M has exactly same number of connected components to M, i.e. n_X. Therefore $L^X_{\lambda^*}$ has at least n_X connected components.
- **(5)** By **(2)** and **(4)**, we conclude that $L^G_{\lambda^*}(Z)$ and $L^X_{\lambda^*}$ do not agree on the number of connected components as long as $n_Z < n_X$.

In this proof, n_Z corresponds to the maximal 0-th Betti number of $L^Z_{\lambda^*}$, i.e. the number of generators of $H_0(L^Z_{\lambda^*})$. If we keep using a mixture of Gaussians as latent vector distribution, all components of $L^Z_{\lambda^*}$ are contractible, so we may use 0 as the maximal i-th Betti number.

Also, n_X corresponds to the 0-th Betti number of M and it worked as the minimal 0-th Betti number of $L^X_{\lambda^*}$. The condition on the class-wise distance of M is used to ensure n_X to be a lower bound. Combining these observations, we can get the following generalized statement.

Theorem 3. Let $\mathcal{D}_Z$ be a mixture of multivariate Gaussian distributions, and let $\mathcal{D}_X$ be the target distribution from data-generating manifold M. Let n_i be the i-th Betti number of M.

Consider a generative model G is used to approximate $\mathcal{D}_X$ using the latent vectors sampled from $\mathcal{D}_Z$. Assume that G is a homeomorphism from $\mathbb{R}^n$ to itself. Assume that data-generating manifold satisfies the conditions of the Theorem 1, and let λ^* be the threshold value that $L^X_{\lambda^*}$ corresponds

to that superlevel set. Assume that for some $j > 0$, the homomorphism ι^* induced by the inclusion $\iota : M \to N_{\delta^*}(M)$ is injective. [5]

If $0 < n_j$, $L^X_{\lambda^*}$ and $L^{G(Z)}_{\lambda^*}$ do not agree on the number of connected component.

Proof. Since $L^X_{\lambda^*}$ is the results of Theorem 1, it includes M and is included by δ^*-neighborhood $N_{\delta^*}(M)$ of M. Define inclusions ι_1, ι_2 as,

- $\iota_1 : M \to L^X_{\lambda^*}$
- $\iota_2 : L^X_{\lambda^*} \to N_{\delta^*}(M)$

Clearly, $\iota = \iota_2 \circ \iota_1$.

Let ι_1^* and ι_2^* be induced homomorphisms of ι_1 and ι_2, resp.

By the assumption, any generator $[a]$ in $H_j(M)$ is injectively mapped to a nonzero generator $\iota^*([a])$ in $H_j(N_{\delta^*}(M))$. Therefore, the j-th Betti number of $N_{\delta^*}(M)$ is equal to that of M, i.e. n_j. Note that j-th Betti number is the rank of j-th homology group $\mathrm{rank}(H_j(N_{\delta^*}(M)))$ Because ι_2^* is a homomorphism from $H_j(L^X_{\lambda^*})$ to $H_j(N_{\delta^*}(M))$, $\mathrm{rank}(L^X_{\lambda^*}) \geq \mathrm{rank}(H_j(N_{\delta^*}(M)))$. Therefore the j-th Betti number of $L^X_{\lambda^*}$ is at least n_j.

However, since $\mathcal{D}_Z$ is a mixture of Gaussians, for any value of λ (including the special case $\lambda = \lambda^*$), L^Z_λ does not have any generator of j-th homology group, so it has j-th Betti number 0 for all $j > 0$. Since G is homeomorphic, it preserves all the Betti numbers, thus $L^{G(Z)}_{\lambda^*} = G(L^Z_{\lambda^*})$ has the same j-th Betti number. As $0 < n_j$, $L^X_{\lambda^*}$ and $L^{G(Z)}_{\lambda^*}$ can never agree on the number of connected components. □

Later in Section 5.2, we will see the Figure 3i from the circles dataset, which is a remarkable example that $L^G_\lambda(Z)$ has the same number of connected components, but does not have any loop (non-contractible circle). This is empirical evidence of the Theorem 3, so it is explained by mismatches in the topology of distributions. Each concentric circle has $\mathbb{Z}$ as its first homology group as circle contains exactly one generator. However, latent vector distribution always has a trivial first homology group, as any superlevel set of a mixture of Gaussians is a set of contractible connected components.

D.5 Details of INC implementations in the Section 5

INC implementation. We start from introducing the optimization for the ideal INC projection when the data-generating manifold M is available.

$$\mathbf{x}^* = \arg\min_{\mathbf{x} \in M} d(\mathbf{x}, \hat{\mathbf{x}}) \tag{11}$$

where d is a metric defined on the domain X. If perfect classification on M is assumed (model is well-trained on M) and $\hat{\mathbf{x}}$ is close enough to the manifold of correct label, classification $f(\mathbf{x}^*)$ is likely to be correct, since $\mathbf{x}^*$ is likely to lie on the correct manifold. Since the data-generating manifold M is unknown, the INC approach runs the following optimization with before the classification.

$$\mathbf{x}^* = G(\mathbf{z}^*) \text{ where } \mathbf{z}^* = \arg\min_{\mathbf{z} \sim \mathcal{D}_Z} d(G(\mathbf{z}), \hat{\mathbf{x}}) \tag{12}$$

where d is a metric defined on the domain X.

When INC is implemented with a reversible generative model G, for any given $\hat{\mathbf{x}} \in \mathbb{R}^n$ there exist a trivial solution $\mathbf{z}^* = G^{-1}(\hat{\mathbf{x}})$ to the optimization (12), achieving $d(G(\mathbf{z}^*), \mathbf{x}) = 0$. This is even true for $\hat{\mathbf{x}}$ out of the manifold, resulting in the situation that the output $\mathbf{x}^* = G(\mathbf{z}^*) = \hat{\mathbf{x}}$ is still out of the data-generating manifold.

To manage this problem, we add another term penalizing a low density of latent vector to the objective function. Thus, in our INC implementation, we solve the following optimization problem.

$$\mathbf{x}^* = G(\mathbf{z}^*) \text{ where } \mathbf{z}^* = \arg\min_{\mathbf{z} \sim \mathcal{D}_Z} [d(G(\mathbf{z}), \hat{\mathbf{x}}) + \alpha(M - p_Z(\mathbf{z}))] \tag{13}$$

[5] Any generator of the j-th homology group $H_j(M)$ of M is mapped to a nonzero generators of the j-th homology group $H_j(N_{\delta^*}(M))$ of δ^*-neighborhood of M.

where α is the regularization factor and M is the maximum possible value of the density p_Z of the latent vector distribution. For the choice of regularization factor, we used the same value $\alpha = 1$ during the entire experiments.

To solve each optimization problem, we used the built-in adam optimizer (Kingma & Ba, 2014) in Tensorflow package, 100 iteration using learning rate 0.01 with random sampling of $\mathbf{z}$.

When implementing INC using a class-aware generative model, we used the following strategy to improve its robustness.

- As class-aware generative model generates each manifold from each Gaussian component, we first sample initial points from each manifold by randomly choosing latent vectors $\mathbf{z}_1, \ldots, \mathbf{z}_l$ from each Gaussian component.
- We run INC for i-th manifold by solving the following optimization.

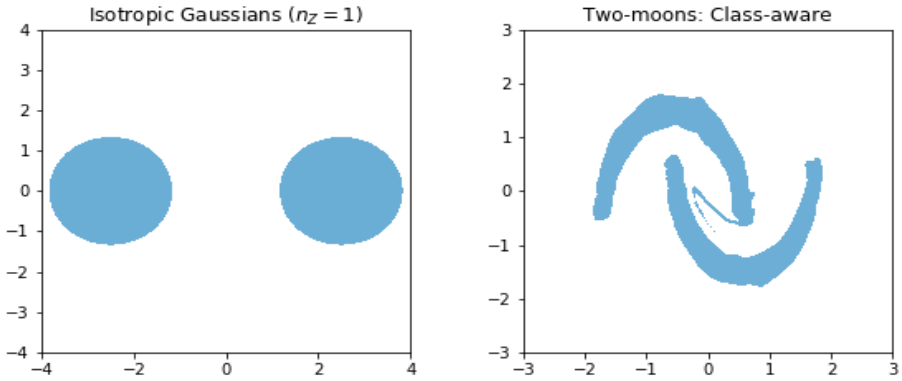
$$\mathbf{x}_i^* = G(\mathbf{z}_i^*) \text{ where } \mathbf{z}_i^* = \arg\min_{\mathbf{z} \sim \mathcal{D}_Z} \left[d(G(\mathbf{z}), \hat{\mathbf{x}}) + \alpha(M_i - p_{Z,i}(\mathbf{z})) \right]$$

  where M_i is the maximum value of i-th Gaussian component. The regularization term is desigend to penalize $\mathbf{z}$ that is unlikely to be generated by i-th Gaussian component, so we only search in the range of i-th Gaussian component, i.e. i-th manifold.
- Choose the final solution $\mathbf{x}_i^*$ achieving the minimum $d(\mathbf{x}_i^*, \hat{\mathbf{x}})$, breaking ties randomly.

Since each search is performed only on each submaifold, the artifact observed in 5.3 never appears during the optimization process. Also, choosing initial points from each manifold prevents the bad initialization problem mentioned in 5.3.

D.6 Discussion about the limitation of topological information

Given a sufficient number of connected components in the latent vector distribution, does class-aware training suggested in this paper end up with a generative model that achieves manifold separation? For this question, the answer is **no**, and the manifold separation depends on other factors, e.g. alignment of latent vector distribution, choice of training parameter, etc.



(a) superlevel set of $\mathcal{D}_Z$

(b) superlevel set of $\mathcal{D}_{G(Z)}$

Figure 7: Failure cases of class-aware training.

Figure 7b shows the superlevel set of $\mathcal{D}_{G(Z)}$ from a class-aware training to learn `two-moons` dataset, when latent vector distribution is a mixture of two Gaussian distributions aligned horizontally (Figure 7a). It is clear that in this case, the generative model induced a connection artifact even when the class-aware training was used.

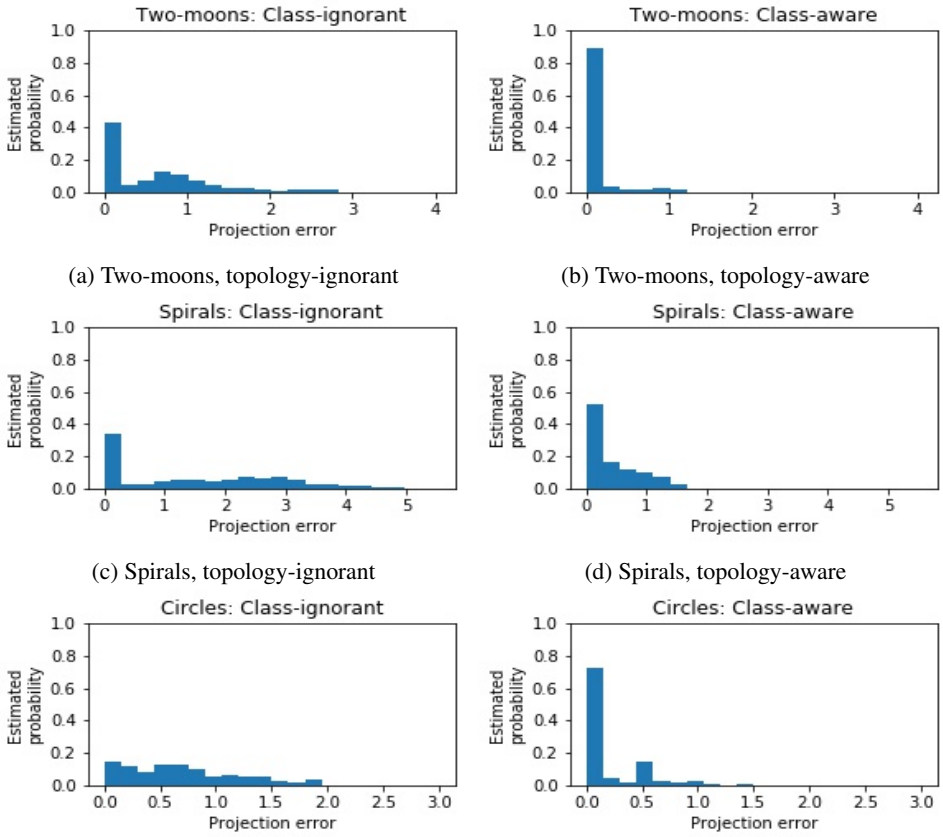
We explain this by interpreting reversible generative models as dynamical systems (Weinan, 2017; Chen et al., 2018; Grathwohl et al., 2018; Zhang et al., 2018). To elaborate, a reversible generative model can be viewed as a dynamical system moving the latent vector distribution to the target

distribution continuously in time. When two Gaussian mixtures are aligned vertically, a reversible generative model is likely to learn how to move the upper (and lower) Gaussian distribution towards the upper moon (and the lower moon, respectively), without being affected by the entanglement of two moons. However, moving the left (and right) Gaussian distribution towards the left moon (and the right moon, respectively) continuously in time is required to avoid the entanglement of two moons during the transition. This case alludes that, information about the topological properties may not be enough to learn a generative model separating manifolds, as it lacks information about how data-generating manifolds are aligned.

E MORE EXPERIMENTAL RESULTS

We present more experimental results about the INC performance comparing topology-aware generative model to its topology-ignorant counterpart.

Histogram for projection error distributions in 5.4. Figure 8 presents the histogram of the projection errors distributed from 0 to the diameter of the distribution. Each row corresponds to each dataset, whereas the first column and the second column represent the results from the topology-ignorant model and the topology-aware model, respectively. All histograms are normalized so that the sum of values adds up to 1. To explain, the y-axis of each histogram is the estimated probability that INC achieves the projection error on the x-axis. Not only that we can observe the improved mean of projection errors in the histograms, but we can also check the reduced standard deviation, i.e. we get more consistent projection errors near the mean.



(a) Two-moons, topology-ignorant

(b) Two-moons, topology-aware

(c) Spirals, topology-ignorant

(d) Spirals, topology-aware

(e) Circles, topology-ignorant

(f) Circles, topology-aware

Figure 8: Histograms of the projection errors of INC. Each y-axis represents the estimated probability that INC incurs the projection error on the corresponding x-axis.