ISOMETRIC REGULARIZATION FOR MANIFOLDS OF FUNCTIONAL DATA

Hyeongjun Heo[1], **Seonghun Oh**[2], **Jae Yong Lee**[3], **Young Min Kim**[1*], **Yonghyeon Lee**[4*]
[1]Seoul National University [2]Yonsei University [3]Chung-Ang University
[4]Korea Institute for Advanced Study
{heo0224, youngmin.kim}@snu.ac.kr
rendell@yonsei.ac.kr jaeyong@cau.ac.kr ylee@kias.re.kr

ABSTRACT

While conventional data are represented as discrete vectors, Implicit Neural Representations (INRs) utilize neural networks to represent data points as continuous functions. By incorporating a shared network that maps latent vectors to individual functions, one can model the distribution of functional data, which has proven effective in many applications, such as learning 3D shapes, surface reflectance, and operators. However, the infinite-dimensional nature of these representations makes them prone to overfitting, necessitating sufficient regularization. Naïve regularization methods – those commonly used with discrete vector representations – may enforce smoothness to increase robustness but result in a loss of data fidelity due to improper handling of function coordinates. To overcome these challenges, we start by interpreting the mapping from latent variables to INRs as a parametrization of a Riemannian manifold. We then recognize that preserving geometric quantities – such as distances and angles – between the latent space and the data manifold is crucial. As a result, we obtain a manifold with minimal intrinsic curvature, leading to robust representations while maintaining high-quality data fitting. Our experiments on various data modalities demonstrate that our method effectively discovers a well-structured latent space, leading to robust data representations even for challenging datasets, such as those that are small or noisy.

1 INTRODUCTION

Implicit Neural Representations (INRs), often referred to as Neural Fields, are functional representations of data points typically expressed as $f_\theta : \mathcal{X} \to V$ with network parameters θ, where $\mathcal{X}$ is an input space such as spatial or temporal coordinates and V is an output vector space (Xie et al., 2022). One significant advantage of this representation is that, since individual data points are continuous functions, one can sample values of V at an arbitrary resolution from $\mathcal{X}$. Moreover, given multiple instances of functions, it has recently been shown that, instead of learning separate network weights for each data instance, a shared neural network can be constructed by conditioning on an auxiliary latent variable z:

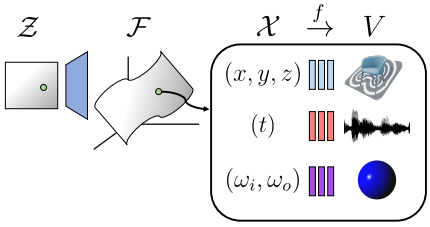

Figure 1: Illustrations of manifolds of functional data.

$$F : \mathcal{X} \times \mathcal{Z} \to V, \tag{1}$$

where, for an instance of a latent vector $z \in \mathcal{Z}$, $F_z := F(\cdot, z)$ corresponds to a data point of functional representation. This latent variable model has been widely applied to tasks such as data compression, generation, prediction, and solving differential equations, across various data types, including images (Sitzmann et al., 2020; Tancik et al., 2020), 3D shapes (Park et al., 2019), textures (Fan et al., 2022), and functions for differential equations (Lu et al., 2019; 2021).

*Corresponding authors.
Project website: heo0224.github.io/IRMF_projectpage

To achieve strong performance in downstream tasks such as interpolation and reconstruction, the latent variable model F must satisfy two essential criteria: (i) it should accurately fit the data instances, resulting in high-fidelity reconstructions, and (ii) the latent space Z should be well-behaved, ensuring that small changes in the latent space lead to gradual and predictable changes in the output. Without proper regularization, particularly when the training dataset is small or noisy, simply fitting the data instances can cause the latent space to become ill-behaved, leading to poor representations and weaker performance in downstream tasks. Recently, Lipschitz regularization (Liu et al., 2022) proposes to learn a well-behaved latent space with a continuous function by promoting smoothness in F. However, they regularize across both input domains, X and Z, and may result in over-smoothing of data instances in X, failing to meet the aforementioned two criteria simultaneously.

In this paper, we propose a balanced regularization method that produces a well-behaved latent space Z without overly smoothing the data in X. Our approach begins with a geometric interpretation of the latent variable model F as a Riemannian manifold of functional data, where the latent variable z corresponds to the coordinates of the manifold, as depicted in Figure 1. Based on this interpretation, we regularize F in the z component to produce a smooth and parsimonious manifold – resulting in small intrinsic curvatures (Lee & Park, 2023) and well-behaved latent spaces – while leaving the x component unregularized, thus preserving the fidelity of individual data instances.

Specifically, we adopt the recent isometric regularization method developed for finite-dimensional data (Lee et al., 2022b), which preserves geometric quantities such as distances and angles between the latent space and the manifold, and extend it to infinite-dimensional functional data. While regularizing the manifold of discretized data on a grid, like images, has been widely studied (Chen et al., 2020; Lee et al., 2021; 2022b; Jang et al., 2023; Lee et al., 2022a; Nazari et al., 2023; Lee & Park, 2023), applying these methods to latent variable INRs is challenging due to variability in sample locations and numbers, making consistent regularization difficult across different data instances. To address this, we propose a discretization-agnostic approach to isometric regularization for F. One notable difference from the finite-dimensional cases is that this approach involves integrating with respect to a positive measure in X, which requires significant computation. To make this practical, we develop an approximate yet efficient algorithm, using methods such as Hutchinson's trace estimator and Monte Carlo approximation.

Through extensive experiments, we validate the effectiveness of our isometric regularization for functional data on various tasks with INRs including neural Signed Distance Functions (SDFs) (Park et al., 2019), neural Bidirectional Reflectance Distribution Functions (BRDFs) (Fan et al., 2022), and Deep Operator Networks (DONets) (Lu et al., 2019) showing that our method is modality-independent. Further, we illustrate that isometric regularization guides the model F to learn an accurate manifold with smooth latent space leading to good generalization performance and robustness to noise in data.

2 Related Work

Latent Variable Models for Functional Data. Recently, many works have shown that continuous signals can be efficiently modeled as a function parameterized by a neural network, referred to as Implicit Neural Representations (INRs). INRs directly map the input variable (i.e., spatial coordinate or time index) into the corresponding value and efficiently represent a broad class of high-resolution data including images (Ha, 2016), audios (Sitzmann et al., 2020), videos (Chen et al., 2022; Li et al., 2021), 3D shapes with SDF (Park et al., 2019), occupancy (Mescheder et al., 2019), and radiance fields (Mildenhall et al., 2020). INRs are also actively employed to represent solutions to diverse differential equations in physics-informed machine learning (Raissi et al., 2019; Karniadakis et al., 2021; Lu et al., 2019; 2021). Instead of representing a single function, recent INR works have shown learning the distribution of a range of functional data by conditioning a shared neural network on a latent variable z for each functional data (Chen & Zhang, 2019; Park et al., 2019; Mescheder et al., 2019; He et al., 2022; Fan et al., 2022; Du et al., 2021). There are two approaches to constructing a latent variable model. One straightforward way to condition is to concatenate input coordinate x with z (Park et al., 2019; Fan et al., 2022; He et al., 2022). Alternatively, novel network architectures are used to condition the latent variable such as hypernetworks (Du et al., 2021; Gokbudak et al., 2024; Lee et al., 2023) or attention networks (Rebain et al., 2022). We emphasize that our framework is generic and can be applied to both architectures.

Geometric Regularization for Manifold Learning. For nite-dimensional data, one can interpret the learned latent space of generative models as an explicit parametrization of the data manifold (Arvanitidis et al., 2017; Lee, 2023). It has been widely demonstrated that proper regularization on latent space can signi cantly improve the performance on downstream tasks for generative models, such as clustering, interpolation, retrieval, and more (Chen et al., 2020; Lee et al., 2021; 2022b; Jang et al., 2023; Lee et al., 2022a; Nazari et al., 2023; Lee & Park, 2023; Lim et al., 2024; Lee, 2024).

However, it is less explored to extend such manifold interpretation of latent variables beyond nite-dimensional data. While Du et al. (2021) view the latent variable model as a manifold of INRs and presents a local isometry loss, they require functions to be sampled from xed points in coordinates with a xed resolution. However, functional data often needs different sampling strategies for each data instance as it considerably affects the tting quality (Park et al., 2019; Sztrajman et al., 2021). Furthermore, it is impossible to sample from xed points with xed numbers for real data in certain scenarios, for example, mobile sensors such as oating buoys change their location over time and some sensors can malfunction (Luo et al., 2024). Instead, we design geometric regularization method speci cally for functional data without xing sampling condition.

3 Riemannian Manifold of Functional Data

In this section, we introduce a geometric view that considers latent variable INRs as a Riemannian manifold of functional data, laying the foundation for subsequent isometric regularization. We start with a formal de nition of the space of functional data $f : X \to V$, where $X = \mathbb{R}^n$ and V is a vector space with the standard inner product, such that the inner product between $v, w \in V$ is $v^T w$. Throughout, we will consider a set of square-integrable mappings

$$F := \{f : X \to V \mid \int f(x)^T f(x)\, dx < \infty\} \tag{2}$$

as a functional data space, which is an in nite dimensional vector space[1].

3.1 Inner Products on Functional Data Space

First, we de ne the inner product on the functional data space F. Let α_1, α_2 be square-integrable functions with a countably additive measure μ (e.g., probability measure) on X. A standard way to de ne an inner product between them is as follows:

$$\langle \alpha_1, \alpha_2 \rangle := \int \alpha_1(x)^T \alpha_2(x)\, d\mu(x). \tag{3}$$

This inner product does not depend on a functional data point $f \in F$. However, in some cases, we need to consider different measures for each functional data point, as important regions in X could be different depending on f. For example, DeepSDF (Park et al., 2019) utilizes truncated SDF (TSDF) that clamps distance values of regions far from the surface to be constant, which means that function values only from areas close to the surface have meaningful information. In such a case, we would like to use a measure concentrated near the surface depending on f. Therefore, in this work, we consider the inner product to depend on the function f by employing a function-dependent measure μ_f,

$$\langle \alpha_1, \alpha_2 \rangle_f := \int \alpha_1(x)^T \alpha_2(x)\, d\mu_f(x) \quad \text{for } f \in F. \tag{4}$$

3.2 Riemannian Manifold Embedded in Functional Data Space

Given our de nition of the functional data space F in Equation (2), we can interpret the model $F : X \times Z \to V$ as a mapping from the latent space Z to the functional data space F:

$$h : Z \to F \quad \text{s.t. } z \mapsto h(z) := F_z, \tag{5}$$

where F_z is a functional data and $F_z(x) := F(x, z)$. Taking DeepSDF as an example, a latent code z is mapped to a 3D shape represented by SDF $F_z : \mathbb{R}^3 \to \mathbb{R}$, which takes a spatial point

[1] F is a Banach space, a complete normed vector space, with respect to the Lebesgue measure.

$x \in X = \mathbb{R}^3$ as input and outputs the signed distance value $v \in V$. Moving z in the latent space leads to a continuous change in the output signed distance $F_z(x)$.

If Z is an m-dimensional manifold (e.g., $\mathbb{R}^m$), then the image of h, denoted by $h(Z) = F(\cdot, Z)$, is an m-dimensional manifold embedded in the functional data space $\mathcal{F}$ under some mild conditions[2], as illustrated in Figure 1. We note that, although F is in finite-dimensional, the embedded manifold is finite-dimensional. This manifold interpretation offers a geometric understanding of the model F, allowing us to leverage tools from differential geometry to develop regularization in a principled manner, as we demonstrate in the subsequent section.

We conclude this section by proposing a Riemannian geometric structure for the embedded manifold. Given the function-dependent inner product $\langle \cdot, \cdot \rangle_f$ for the functional data space $\mathcal{F}$ in Equation (3), a standard way to define a Riemannian metric for the embedded manifold is via projecting $\langle \cdot, \cdot \rangle_f$ to the manifold. If $Z = \mathbb{R}^m$ is treated as a local coordinate space, then the projected Riemannian metric can be expressed in Z as follows:

$$ds^2 = \sum_{i,j=1}^{m} \int_X \frac{\partial F(x;z)}{\partial z^i}^T \frac{\partial F(x;z)}{\partial z^j} d\mu_{F_z}(x) dz^i dz^j; \tag{6}$$

where the integral $h_{ij}(z) := \int_X \frac{\partial F(x;z)}{\partial z^i}^T \frac{\partial F(x;z)}{\partial z^j} d\mu_{F_z}(x)$ forms an $m \times m$ positive-definite matrix $H(z) = (h_{ij})_{i,j=1}^m$ called a Riemannian metric. We note that, with this metric, given an infinitesimal change in the latent value $z \mapsto z + dz$, the length of dz is computed via $dz^T H(z) dz = \int_X (dF(x;z))^T dF(x;z) d\mu_{F_z}(x)$, where $dF(x;z) \triangleq F(x;z+dz) - F(x;z)$.

4 Isometric Regularization

Without proper regularization, the latent space can become ill-behaved, over fitting to the data instances, which is exacerbated by the infinite dimensionality of the function space. Our goal of finding a well-behaved latent space without compromising data fidelity can be restated in geometric terms: the given data instances lie in the manifold $h(Z)$ while the mapping h from Z to the manifold $h(Z)$ should preserve geometry as much as possible. The former condition of fitting data instances is enforced by a task-relevant loss function, which varies depending on the application. Examples in the experiment section include the signed distance and the rectance fitting loss in deep generative models and the regression loss in operator learning. The latter geometry-preserving condition can be achieved through our proposed isometric regularization.

By `geometry-preserving,' we mean that distances and angles measured in the manifold should be preserved in the latent space. Consequently, slight changes in latent values will not result in dramatic changes in the output data but will cause changes of a consistent magnitude, leading to a well-behaved latent space. Formally, let the latent space $Z = \mathbb{R}^m$ be assigned with the identity metric and the manifold of functional data be assigned with the projected Riemannian metric in Equation (6). Throughout, we will denote the Jacobian of F by $J(x;z) = \frac{\partial F(x;z)}{\partial z} \in \mathbb{R}^{\dim(V) \times m}$ and the projected metric

$$H(z) = \int_X J(x;z)^T J(x;z) d\mu_{F_z}(x) \in \mathbb{R}^{m \times m}. \tag{7}$$

A mapping h is called an isometry if $I = H(z)$ for all $z \in Z$. If F satisfies the above condition, then for infinitesimal changes in the latent value dz and the corresponding change in the data $dF(x;z) \triangleq F(x;z+dz) - F(x;z)$, the norm of dz in the latent space is equal to the norm of dF, i.e., $dz^T dz = dz^T H(z) dz = \int_X dF^T dF d\mu_{F_z}(x)$, meaning that F preserves local distances and angles. If this holds for all z, then F preserves the global geometry. As discussed in Lee et al. (2022b), we encourage a scaled isometry for F that preserves angles and scaled distances, such that $cI = H(z)$ for some positive scalar c and all z.

It is important to note that enforcing this scaled isometry condition does not impose a smoothness requirement on the x-coordinates, unlike Lipschitz regularization. For instance, consider the limiting

[2] The mapping $h : Z \to \mathcal{F}$ is an embedding if it is injective immersion. In other words, (i) (injectivity) if $F(x;z_1) = F(x;z_2)$, then $z_1 = z_2$ for all $x \in X$ and (ii) (immersion) if $\frac{\partial F(x;z)}{\partial z} v = 0$ for $v \in \mathbb{R}^m$, then $v = 0$ for all x, z.

case where (i) $\lambda = cH(z)$ is exactly satisfied for all z and (ii) $d_{F_z} = d$ is independent of z. In this scenario, a trivial solution is that $J(x;z)$ remains constant over z. Crucially, no conditions are imposed on the x-coordinates. This is one of the key properties of our regularization, which results in a well-behaved latent space without compromising data fidelity.

Relaxed Distortion Measure. We introduce a functional called the relaxed distortion measure to impose isometric regularization for manifolds of functional data. It quantifies how far a mapping F is from being a scaled isometry. To handle the continuous data space, we apply an approximation based on efficient sampling algorithm. Here, we provide formulas for the relaxed distortion measure with minimal details; please refer to Appendix A for more information and proofs. Throughout, we will use a probability measure for F_z in Equation (7) and denote its density function by $p(x;F_z)$. Then the Riemannian metric can be written as $H(z) = E_{x\ p(x;F_z)}[J^T(x;z)J(x;z)] \in \mathbb{R}^{m\ m}$. Let $_i(z)$ be the eigenvalues of $H(z)$ for $i = 1;\dots;m$.

An equivalent characterization of the scaled isometry condition exists: if $_i(z) = c$ for some positive scalar c and all i and z, then F is a scaled isometry. Enforcing this condition for all $z \in \mathbb{R}^m$ is unnecessary since latent points only occupy a specific region. We introduce a latent probability density P_Z and focus on its support. Then, a coordinate-invariant relaxed distortion measure $G(F;P_Z)$ for F with respect to P_Z can be defined as follows:

$$G(F;P_Z) := E_{z\ P_Z}\Big[\sum_{i=1}^{m} \frac{P_{\ i}(z)}{E_{z\ P_Z}[\sum_{i=1}^{m}\ _i(z)=m]} \quad 1\ ^2\Big]; \tag{8}$$

which is minimal if and only if F meets the scaled isometric condition in support of P_Z. Ignoring a constant additive term, $G(F;P_Z)$ becomes a scalar multiple of

$$\frac{E_{z\ P_Z}[\mathrm{Tr}(\ E_{x\ p(x;F_z)}[J^T(x;z)J(x;z)]\ ^2)]}{E_{z\ P_Z}[\mathrm{Tr}(\ E_{x\ p(x;F_z)}[J^T(x;z)J(x;z)])]^2}. \tag{9}$$

When training manifolds of functional data F, we add the term above to the original loss term to incorporate isometric regularization for the latent mapping. The expression in Equation (9) is similar to the one introduced in Lee et al. (2022b), but ours involves $E_{x\ p(x;F_z)}$ in the traces, and the Jacobian also depends on x. This difference makes applying existing implementations non-trivial, motivating us to develop a new approximate and efficient distortion computation algorithm.

Efficient Approximation of Eq. (9). The computation of Equation (9) involves (i) calculating the Jacobian $J(x;z)$ and (ii) computing the expectation of J^TJ with respect to $p(x;F_z)$. These calculations must be performed at each training iteration, which significantly slows down the overall process. To address this, we introduce several practical techniques for approximate computation. First, we estimate the trace terms using Hutchinson's stochastic trace estimator (Hutchinson, 1989): $\mathrm{Tr}(A) = E_{v\ N(0;I)}[v^TAv]$, which allows us to bypass the need to compute the full Jacobian $J(x;z)$ and instead focus on Jacobian-vector and vector-Jacobian products. Second, to compute the expectation of J^TJ with respect to $p(x;F_z)$, we typically need to sample from $p(x;F_z)$. However, the sampling distribution $p(x;F_z)$ evolves during the training of F, making the online sampling infeasible. To mitigate this, we use of fline samples from $p(x;F_i)$, where F_i represents the functional data for training z_i.

Algorithm 1: Efficient approximation of Eq. (9)

Precondition: input concatenation $(F : \mathbb{R}^{n+m} \to \mathbb{R}^l)$
Input: latent codes $\{z_0;\dots;z_N\}$ & input coordinate samples $\{\{x_0^{(0)};\dots;x_0^{(K)}\};\dots;\{x_N^{(0)};\dots;x_N^{(K)}\}\}$
Output: Relaxed distortion measure G

```
G_1; G_2  0
Augment z with the modified mix-up data-augmentation
forall z_i in z do
    x_i  {x_i^(0); ...; x_i^(K)}
    Sample vector v_i  N(0; I_{m m})
    Expand v_i by repeating K times
    Augment vector v_i by concatenating [0_{k n}; v_i]
    Compute G = J(x_i; z_i) v_i with Jacobian-vector product
    G_1  G_1 + E_z[E_x[G^T G]]
    Compute D = G^T @F(x_i; z_i)/@(x; z) with vector-Jacobian product
    Slice the index of D by taking the last m-th components
    G_2  G_2 + E_z[E_x[D]^T E_x[D]]
end
G  G_2 / G_1
return G
```

We also apply the latent augmentation method from Chen et al. (2020); Lee et al. (2022b) to define P_Z. A pseudocode for the approximate computation of the relaxed distortion measure is provided in Algorithm 1; more details can be found in Appendix B.

Figure 2: Toy example of neural SDFs with five circles. (Left) Two-dimensional latent space of auto-decoder trained with and without isometric regularization. (Right) SDFs decoded from linear interpolants & L1 errors on SDFs.

5 EXPERIMENTS

In this section, we conduct extensive experiments to show the effect of isometric regularization on three data modalities: neural SDFs, neural BRDFs, and neural operators.

5.1 NEURAL SDFS

SDF is a function $f : \mathbb{R}^n \to \mathbb{R}$ that maps a spatial point to its signed distance to the surface of the shape. The surface points of the shape are extracted from the zero-level set of f. We train the neural SDFs with the latent variable model, denoted by $F : X \times Z \to \mathbb{R}$, where $x \in X$ represents spatial coordinates, $z \in Z$ is a shape latent vector, and $F(\cdot; z)$ is a SDF value, following the auto-decoder in DeepSDF (Park et al., 2019). We provide experimental details in Appendix C.1.

5.1.1 TOY EXAMPLES ON SIMILAR SHAPES

We first demonstrate the effect of isometric regularization with simple 2D shapes. We train auto-decoders using a two-dimensional latent space $z \in \mathbb{R}^2$ without any regularization (AD) and with isometric regularization (IsoAD), and compare the results. Appendix D.1 contains additional analysis on the probability density $p(x; F_z)$.

Table 1: L1 errors ($\times 10^{-3}$) on linear interpolation.

	N	AD	IsoAD (Ours)
Circle	5	36.5885	2.7497
	10	10.1056	2.5622
	20	9.5139	1.9630
Square	5	13.1302	6.7625
	10	2.2039	1.9842
	20	2.7227	1.4770
Triangle	5	21.8526	4.4000
	10	2.0579	1.7629
	20	2.7957	1.3722

Dataset. We make training datasets with three kinds of simple shapes: circles, squares, and equilateral triangles. Each dataset contains N similar shapes whose parameters change linearly, for example, circles with radius from 0.1 to 0.5 or squares with side lengths from 0.1 to 0.5. Then, we train the network with the SDF values of each shape sampled from a 32×32 grid.

Results. Given the smallest and the biggest shapes for each model, we generate intermediate shapes by linearly interpolating the latent space. We evaluate the decoded intermediate shapes against reference shapes created by interpolating the shape parameters. We compute the L1 errors at 256×256 grid points. Quantitative results in Table 1 show that isometric regularization significantly improves the interpolation results, especially when the number of training data N is small. Figure 2 shows the latent space of auto-decoders trained with five circles. When we linearly increase the circle's radius, the most intuitive latent manifold would be a straight line with equidistant latent codes. While auto-decoder without regularization leads to a distorted latent space resulting in undesirable interpolation results even with extremely simple shapes, isometric regularization encourages the output shape to expand with constant velocity leading to predictable interpolation results.

Table 2: Quantitative results on surface reconstruction.

Dataset	MNIST				ShapeNet			
N	100		500		271		542	
Metrics	L1 error average	L1 error median	L1 error average	L1 error median	CD average	CD median	CD average	CD median
DeepSDF	0.0366	0.0355	0.0288	0.0285	0.002202	0.001812	0.001588	0.001207
LipDeepSDF	0.0351	0.0341	0.0279	0.0255	0.001999	0.001606	0.001361	0.001054
IsoDeepSDF (ours)	0.0283	0.0265	0.0267	0.0245	0.001915	0.001472	0.001263	0.000925

5.1.2 SURFACE RECONSTRUCTION

We now train neural SDFs with a variety of more complex 2D & 3D shapes from MNIST (Deng, 2012) and ShapeNet (Chang et al., 2015). Similar to Section 5.1.1, we use auto-decoder architecture for training 2D SDFs. For training the autoencoder with 3D SDFs, we use an encoder from Pointnet Qi et al. (2017) and a decoder from DeepSDF, following the experimental setup from Liu et al. (2022).

We can then reconstruct the surfaces by finding latent codes that minimize the output SDF values at the surface points provided at the test time. Test time optimization of a latent code is commonly used in auto-decoders and autoencoders to reconstruct a surface from the point cloud data (Park et al., 2019; Gurumurthy & Agrawal, 2019; Liu et al., 2022). In particular, latent code optimization is needed when a simple pass forward through the autoencoder leads to unsatisfying results. We compare our method, referred to as IsoDeepSDF, with vanilla DeepSDF without regularization and DeepSDF with Lipschitz regularization (LipDeepSDF) (Liu et al., 2022).

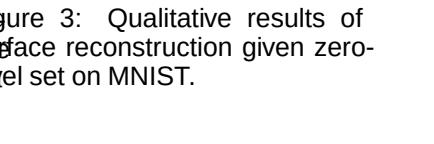

Figure 3: Qualitative results of surface reconstruction given zero-level set on MNIST.

MNIST Dataset. We make two datasets with different numbers of training data, $N = 300; 1500$. The datasets contain 100 & 500 images randomly chosen from each of three digits [6, 8, 9]. The images are transformed to 2D signed distance fields on a 64×64 grid, where the contour of the digit is the zero-level set. For test time optimization, we randomly sample 256 points from zero-level surfaces of the test dataset with 100 images from each digit [6, 8, 9].

ShapeNet Dataset. We randomly choose 5% ($N = 271$) and 10% ($N = 542$) of shapes from the chair category of ShapeNetV2 for training datasets. As we observe that the choice of training dataset strongly impacts the reconstruction results, we make five different datasets for each random choice. Quantitative results are the average of the metrics evaluated from each dataset. The test-time optimization reconstructs the full 3D shape from partial point clouds obtained by deleting the right half of the surface point cloud.

Results. We compute the L1 error for 2D SDFs on a 256×256 grid and Chamfer distance (CD) for 3D shapes with 30,000 points on the surfaces for evaluation metrics. Quantitative results are summarized in Table 2. IsoDeepSDF quantitatively outperforms the others for all datasets, demonstrating that isometric regularization helps to learn a well-behaved latent space, leading to better reconstruction results. While decreasing N severely deteriorates the reconstruction quality on the MNIST dataset, isometric regularization significantly reduces the performance degradation, providing reliable regularization despite scarce data points. Figure 3 and Figure 4 each show the qualitative results of surface reconstruction on the MNIST and ShapeNet datasets. Our method qualitatively shows better reconstruction results for both cases with 2D point clouds and 3D partial point clouds. Especially for 3D shapes, DeepSDF fails to fully reconstruct the unseen parts of the chairs, while our method can better reconstruct the overall shape given partial observations.

Figure 4: Qualitative results of surface reconstruction on ShapeNet chair dataset. Our method helps to learn better latent space, leading to better reconstruction results from partial observation.

5.2 NEURAL BRDFS

The bidirectional re ectance distribution function (BRDF) is a function that takes the incoming and outgoing directions of light as input and outputs the re ected radiance on the surface, which is used to render the appearance of materials. Recent works have proposed generalizable BRDF representations with latent variable models to reconstruct a BRDF of unseen materials from its samples (Sztrajman et al., 2021; Fan et al., 2022; Gokbudak et al., 2024). We show that isometric regularization can complement those methods to learn better latent space for BRDFs, resulting in better reconstruction quality.

We train neural BRDFs with the auto-decoder architecture by concatenating a latent variable z with input directions, similar to Fan et al. (2022). We then optimize the latent codes to reconstruct the entire BRDF of unseen material from BRDF samples. To evaluate the results of BRDF reconstruction, we render a simple scene with the reconstructed BRDF with Mitsuba 3 renderer (Jakob et al., 2022). We compare the results of our method (IsoAD) with vanilla auto-decoder (AD) and auto-decoder with Lipschitz regularization (LipAD). We provide experimental details in Appendix C.2.

Dataset. We use the MERL dataset (Matusik et al., 2003), a common BRDF dataset measured from 100 real isotropic materials. We split the dataset into 80 materials for training and 20 materials for the test dataset. We train the model with various numbers of training data $N = 20; 40; 60; 80$. We make ve different datasets for $N = 20; 40; 60$ with a random choice of materials from the full training dataset ($N = 80$) and compute the average metrics for evaluating BRDF reconstruction.

Results. Figure 6 shows the reconstruction accuracy measured in PSNR and SSIM (Wang et al., 2004). Isometric regularization improves the BRDF reconstruction by a large margin for both metrics. The effect of regularization is prominent when the training data is greatly reduced to 20, maintaining robust results while the reconstruction results of AD drastically degrade. In particular, IsoAD trained with 20 materials shows higher PSNR and SSIM than the baselines trained with the full training dataset of 80 materials. This result demonstrates that isometric regularization enhances the generalization performance without compromising the delity of reconstruction. Qualitative results on BRDF reconstruction are shown in Figure 5. The quality of reconstructed materials aligns with what we expect from quantitative metrics.

5.3 NEURAL OPERATORS

The Deep Operator Network (DONet) (Lu et al., 2019), originally introduced for solving PDEs, is a standard architecture for constructing neural operators that map between functions. In this work, we demonstrate that incorporating isometric regularization can signi cantly improve the performance of operator learning. The neural operator is a regression task that aims to learn a mapping G from an input function $u : X \to \mathbb{R}$ to an output function $o : Y \to \mathbb{R}$, denoted by $G : u(x) \mapsto o(y)$, using pairs of input-output functions. Speci cally, we represent the input as a vectorized function $u_{\text{vec}} = (u(x_1); \dots; u(x_M)) \in U$ for a set of xed points $\{x_k\}_{k=1}^M$. We denote the training dataset of size N by $\{(u_{\text{vec},i}; o_i(y))\}_{i=1}^N$.

Figure 5: Qualitative results on BRDF reconstruction.

Figure 6: Average PSNR (left) and SSIM (right) on BRDF reconstruction.

Our model consists of two networks: an encoder $E : U \to Z$ and a decoder $F : Z \times Y \to \mathbb{R}$, where the output function is modeled as $s(y) = F(E(u_{vec}); y)$ for an input u_{vec}. When u_{vec} is high-dimensional, E encodes it into a low-dimensional vector that captures the essential information required to predict the output function. This structure allows F to be interpreted as a latent variable model for functional representations, enabling the effective application of isometric regularization. We train E and F by minimizing $\sum_{i,j} \|o_i(y_j) - F(E(u_{vec,i}); y_j)\|^2$ for a set of query points $\{y_j\}_{j=1}^{M'}$ and evaluate the regression performance on the test dataset. To assess the robustness of the model, we introduce varying levels of noise to the output functions in the training dataset, while keeping the input functions clean, simulating real-world uncertainties such as measurement errors or numerical approximation inaccuracies. We compare the performance of three models: the unregularized DONet, the model with Lipschitz regularization (LipDONet), and the one with isometric regularization (IsoDONet).

Dataset. This study focuses on two types of PDE datasets: the reaction-diffusion equation, as discussed in Yang et al. (2022), and the Darcy flow problem, based on Lu et al. (2022). The reaction-diffusion equation describes how a solution $u(t, x)$ evolves over time and space under the effects of diffusion and reaction forces:

$$\frac{\partial}{\partial t} u(t, x) = \frac{\partial^2}{\partial x^2} u(t, x) + ku^2 + f(x), \quad (t, x) \in [0, 1] \times [0, 1] \tag{10}$$

with some initial and boundary conditions. The operator we aim to learn is a mapping $G : f(x) \mapsto u(t, x)$, where $f(x)$ represents the forcing term.

The Darcy flow problem models fluid flow through a porous medium, of which steady state on a unit square is given by:

$$-\nabla \cdot (a(x, y) \nabla u(x, y)) = f(x, y), \quad (x, y) \in [0, 1]^2 \tag{11}$$

with some boundary conditions. The operator we aim to learn is a mapping $G : a(x, y) \mapsto u(x, y)$, where $a(x, y)$ is the diffusion coefficient and $u(x, y)$ is fluid density, while assuming a fixed external force $f(x, y)$. These datasets are generated by solving the PDEs; details regarding the datasets can be found in Appendix C.3.