Manifold Fitting with CycleGAN

Zhigang Yao[a,1], **Jiaji Su**[a,1], **and Shing-Tung Yau**[b,c,1]

This manuscript was compiled on December 13, 2023

Manifold fitting, which offers substantial potential for efficient and accurate modeling, poses a critical challenge in non-linear data analysis. This study presents a novel approach that employs neural networks to fit the latent manifold. Leveraging the generative adversarial framework, this method learns smooth mappings between low-dimensional latent space and high-dimensional ambient space, echoing the Riemannian exponential and logarithmic maps. The well-trained neural networks provide estimations for the latent manifold, facilitate data projection onto the manifold, and even generate data points that reside directly within the manifold. Through an extensive series of simulation studies and real data experiments, we demonstrate the effectiveness and accuracy of our approach in capturing the inherent structure of the underlying manifold within the ambient space data. Notably, our method exceeds the computational efficiency limitations of previous approaches and offers control over the dimensionality and smoothness of the resulting manifold. This advancement holds significant potential in the fields of statistics and computer science. The seamless integration of powerful neural network architectures with generative adversarial techniques unlocks new possibilities for manifold fitting, thereby enhancing data analysis. The implications of our findings span diverse applications, from dimensionality reduction and data visualization to generating authentic data. Collectively, our research paves the way for future advancements in non-linear data analysis and offers a beacon for subsequent scholarly pursuits.

Manifold fitting | deep generative neural network | CycleGAN

Significance Statement

Manifold fitting, a crucial challenge in non-linear data analysis, holds immense potential for efficient and accurate modeling. However, existing methods face difficulty in striking a balance between accurate estimation and computational efficiency. In this study, we harness the adversarial generative network to model the relationship between low-dimensional latent space and high-dimensional ambient space, mirroring Riemannian exponential and logarithmic maps, for robust manifold analysis. Through extensive simulations and real data experiments, we demonstrate the effectiveness of our method in capturing the intricate structure of manifolds embedded within the high-dimensional data. Our approach has profound implications for both statistics and computer science, particularly in domains that demand dimensionality reduction or the processing of non-Euclidean data. By providing a solution that overcomes the limitations of previous methods, our research paves the way for enhanced data analysis and offers valuable insights for diverse applications in the scientific community.

Numerous prevailing statistical methods operate under the assumption of linear dependence among their features. While these methods offer the advantage of simplifying data representation, they entail challenges in accurately capturing more complex patterns, particularly in high-dimensional scenarios where data tend to reside close to low-dimensional manifolds. To overcome these limitations, the integration of more sophisticated manifold-learning techniques becomes imperative.

Manifold-learning techniques, as classified by Yao et al. (1), encompass three distinct categories: *manifold embedding*, *manifold denoising*, and *manifold fitting*. The manifold-embedding methods aim to uncover a low-dimensional representation of high-dimensional data sets while preserving point-wise distances (2–7). In contrast, manifold-denoising methods target the identification and mitigation of outliers within data distributions around low-dimensional manifolds (8–12). But, despite their notable contributions to dimension reduction and various fields, many of these techniques suffer from a dearth of geometric details and a lack of robust theoretical foundations (1).

Manifold fitting, the third category, is a fundamental task that involves reconstructing a smooth manifold to accurately approximate the geometry and topology of a hidden low-dimensional manifold. Noteworthy contributions by Genovese et al. have focused on the minimax risk of manifold fitting under Hausdorff distance, providing valuable insights into effectively handling noise and improving smoothness (13, 14). Furthermore, the utilization of the ridge of the empirical distribution as a proxy has been explored to mitigate the sample size requirements (15–17). However, as noted by Dunson and Wu (18), these kernel-based methodologies overlook the intrinsic geometry of the domain and lead to sub-optimal outcomes with convergence rates dependent on the ambient dimensionality rather than on the intrinsic dimensionality. To address this limitation, several algorithms, including (17, 19–21), have smooth functions incorporated that capture spatial properties, representing the output manifold as its root set or ridge, while some other methods, like (22, 23), utilize the local covariance information to project sample points onto low-dimensional structures. However, these algorithms often entail high computational costs, necessitating further efficiencies and reduce the computational complexity in manifold fitting tasks.

The concept of smooth manifolds is also fundamental in machine learning, especially in computer vision. Dating back to the era of basic image classification,

Author affiliations: [a]National University of Singapore; [b]Tsinghua University; [c]Harvard University

Z.Y. and S.Y. designed research; Z.Y. and J.S. performed research; Z.Y., J.S. and S.Y. wrote the paper.

The authors declare no competing interest.

[1]Z.Y., J.S. and S.Y. contributed equally to this work.

[2]To whom correspondence may be addressed. E-mail: styau@tsinghua.edu.cn or zhigang.yao@nus.edu.sg.

it has illuminated the intricate relationship between images and their defining features. Recently, generative models such as Autoencoders (24) and Generative Adversarial Networks (GANs, (25)), along with their numerous variations, have garnered widespread attention, all underpinned by the manifold hypothesis. These techniques have made considerable strides in navigating distributions adjacent to manifolds, with some even venturing into the domain of manifold denoising within image spaces. Nevertheless, most of them focus on the efficiency of noise reduction, the plausibility of synthesized images, and the results induced by feature modulation. In contrast, the task of reconstructing the intrinsic structures of latent manifolds has been comparatively neglected and represents an essential avenue for future research efforts.

To exploit the remarkable potential of neural networks in fitting smooth functions describing the latent manifold, we propose a novel manifold fitting method based on the adversarial generative framework. This approach is inspired by recent advances in deep generative neural networks, density estimators (26, 27), and manifold estimators (1, 20) and offers several key advantages.

Firstly, utilizing deep neural networks, we can accurately fit smooth functions and generate image sets that provide more precise estimations of the latent manifold, ensuring control over dimensionality and smoothness. Secondly, our proposed method offers intuitive mapping functions between low-dimensional latent space and high-dimensional ambient space, echoing the Riemannian exponential and logarithmic maps, enabling seamless projection of new data points onto the manifold. Lastly, by establishing mappings between spaces, we gain insights into point-wise relationships along the manifold, laying the groundwork for further research. In what follows, we will present our model and substantiate its outstanding performance and broad applicability through extensive experimentation.

Methodology

The framework of our methods centers on viewing the observing distribution as the convolution of a distribution defined on a smooth manifold and a noise distribution within the ambient space. Given that any smooth manifold admits a complete Riemannian metric (as any manifold embeds into some Euclidean space as a closed subset according to the Whitney embedding theorem), the exponential map defined at any chosen point on a connected smooth manifold yields a smooth and surjective map to the manifold. In other words, fitting a manifold can be regarded as learning a smooth function that maps from a low-dimensional Euclidean space to a high-dimensional Euclidean space—a task in which neural networks excel.

To fit such functions and ensure their near-bijective nature, we leverage the fundamental framework of CycleGAN (26) to train the two generators as the exponential map and logarithmic map at a specific point on the latent manifold.

Model and assumption. Assume there are two domains $\mathcal{X}$ and $\mathcal{Y}$, with dimensionality of d and D respectively, where $d < D$. We focus on a random vector $Y \in \mathcal{Y}$, which can be represented as the sum of two components: Z and ξ. Here, $Z \in \mathcal{Y}$ is an unobserved random vector that follows a distribution ω supported on the latent manifold $\mathcal{M}$. Additionally, $\xi \sim \phi_\sigma$ represents the observation noise in ambient space, which is independent of Z and has a standard deviation σ. Consequently, the distribution of Y can be viewed as the convolution $\nu = \omega \star \phi_\sigma$, whose density at point y can be expressed as

$$\nu(y) = \int_{\mathcal{M}} \phi_\sigma(y - z)\omega(z)\,\mathrm{d}z.$$

We assume that $\mathcal{Y}_N = \{y_i\}_{i=1}^N$ is the collection of observed data points, also in the form $y_i = z_i + \xi_i$ for $i = 1, \cdots, N$. Here, (y_i, z_i, ξ_i) corresponds to N independent and identically distributed (iid) realizations of (Y, Z, ξ). Our objective is to estimate the latent manifold $\mathcal{M}$ based on the observations $\mathcal{Y}_N$, and we make the following main assumptions: 1) The ambient space is D-dimensional Euclidean space with the standard norm, i.e. $\mathcal{Y} = \mathbb{R}^D$. 2) The noise distribution ϕ_σ is a Gaussian distribution supported on $\mathcal{Y}$ with density $\phi_\sigma(\xi) = (\frac{1}{2\pi\sigma^2})^{\frac{D}{2}} \exp(-\frac{\|\xi\|_2^2}{2\sigma^2})$. 3) The latent manifold $\mathcal{M}$ is a twice-differentiable and compact d-dimensional submanifold, embedded in $\mathcal{Y}$. 4) The distribution ω is a uniform distribution, with respect to the d-dimensional Hausdorff measure, on $\mathcal{M}$. 5) The intrinsic dimension d and noise standard deviation σ are both known. These assumptions are inherited from previous theoretical works and are subject to potential relaxation. A more detailed discussion can be found in the SI Appendix.

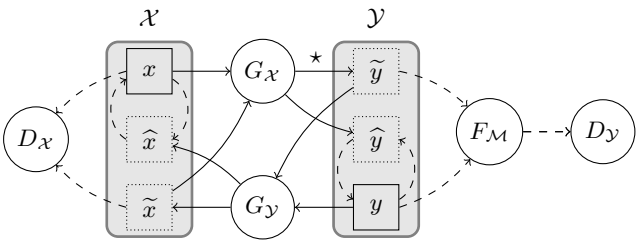


Fig. 1. The schematic representation of our framework. The shaded rectangles depict the two domains, while the circular components symbolize the individual sub-networks. The solid arrows indicate the transfer paths within the generators, while the dashed arrows denote the paths for computing the loss function.

Our target is to learn smooth mappings between the domains $\mathcal{X}$ and $\mathcal{Y}$. The procedure of our method is depicted in Fig. 1. Our model comprises two generators: $G_{\mathcal{X}} : \mathcal{X} \to \mathcal{Y}$ and $G_{\mathcal{Y}} : \mathcal{Y} \to \mathcal{X}$. During the forward propagation step, we collect a batch of real data points x and y. To generate synthetic data in domain $\mathcal{Y}$, denoted as $\widetilde{y}$, we introduce Gaussian noise to $G_{\mathcal{X}}(x)$, that is, $\widetilde{y} = G_{\mathcal{X}}(x) + \xi$. In contrast, in domain $\mathcal{X}$, synthetic data $\widetilde{x}$ is directly obtained through $\widetilde{x} = G_{\mathcal{Y}}(y)$. Subsequently, we calculate the recovered data: $\widehat{x} = G_{\mathcal{Y}}(\widetilde{y})$ and $\widehat{y} = G_{\mathcal{X}}(\widetilde{x})$. In order to differentiate between real and synthetic data points, we introduce two adversarial discriminators, $D_{\mathcal{X}}$ and $D_{\mathcal{Y}}$, aiming to distinguish between $\{x, \widetilde{x}\}$ and $\{y, \widetilde{y}\}$ respectively. Additionally, we incorporate a sub-module $F_{\mathcal{M}}$ performing a pseudo manifold fitting step before passing data to $D_{\mathcal{Y}}$, which helps in aligning the data points with the latent manifold and expedites the training process (1). During the training of the neural network modules, our objective function incorporates two primary components: *adversarial losses* (25) and *cycle consistency loss* (26). These terms serve distinct purposes to ensure effective learning and coherence between $G_{\mathcal{X}}$ and $G_{\mathcal{Y}}$.

Adversarial loss. Adversarial loss, inspired by (25), is employed to align the latent distributions in the two domains. By using discriminators $D_{\mathcal{X}}$ and $D_{\mathcal{Y}}$, we adjust the generated samples, $\widetilde{x}$ and $\widetilde{y}$, to resemble the real samples, x and y, respectively. This adversarial training strategy enables the networks to capture the underlying characteristics and statistical properties of the target domain. To stabilize the training and generate highly credible results, we adopt the least square GAN (LSGAN) loss proposed in (28). The loss function of $G_{\mathcal{X}}$ and its discriminator $D_{\mathcal{Y}}$ can be expressed as

$$\begin{aligned}\mathcal{L}_{\mathcal{X}\to\mathcal{Y}}(G_{\mathcal{X}}, D_{\mathcal{Y}}, F_{\mathcal{M}}) =& \frac{1}{m}\sum_{i=1}^{m}[D_{\mathcal{Y}} \circ F_{\mathcal{M}}(\widetilde{y}_i) - 0]^2 \\ &+ \frac{1}{n}\sum_{i=1}^{n}[D_{\mathcal{Y}} \circ F_{\mathcal{M}}(y_i) - 1]^2,\end{aligned}$$

where m and n are the batched sample sizes from spaces $\mathcal{X}$ and $\mathcal{Y}$, respectively. By removing the $F_{\mathcal{M}}$ sub-module, we introduce a similar adversarial loss for $G_{\mathcal{Y}}$ and $D_{\mathcal{X}}$ as well, i.e. $\mathcal{L}_{\mathcal{Y}\to\mathcal{X}}(G_{\mathcal{Y}}, D_{\mathcal{X}})$.

Cycle consistency loss. Although adversarial training has the potential to learn mappings that yield outputs distributed identically to the target, a network with sufficient capacity can map the same input set to various random permutations of the target set. Consequently, any of the acquired mappings can induce an output distribution that aligns with the target distribution (26). To further reduce the space of possible mappings, we introduce the term cycle consistency loss:

$$\mathcal{L}_c(G_{\mathcal{X}}, G_{\mathcal{Y}}) = \frac{1}{m}\sum_{i=1}^{m}\|x_i - \widehat{x}_i\|_1 + \frac{1}{n}\sum_{i=1}^{n}\|y_i - \widehat{y}_i\|_1,$$

with $\|\cdot\|_1$ being the L^1 norm. By controlling the cycle consistency loss, we can achieve additional objectives such as noise reduction, interpolation, and more, as demonstrated in the sections below.

Full objective. By combining these loss terms, we are able to formulate the comprehensive objective for the model as follows:

$$\begin{aligned}\mathcal{L}(G_{\mathcal{X}}, G_{\mathcal{Y}}, F_{\mathcal{M}}, D_{\mathcal{X}}, D_{\mathcal{Y}}) =& \mathcal{L}_{\mathcal{X}\to\mathcal{Y}}(G_{\mathcal{X}}, D_{\mathcal{Y}}, F_{\mathcal{M}}) \\ &+ \mathcal{L}_{\mathcal{Y}\to\mathcal{X}}(G_{\mathcal{Y}}, D_{\mathcal{X}}) + \lambda\mathcal{L}_c(G_{\mathcal{X}}, G_{\mathcal{Y}}),\end{aligned}$$

where λ is a negative parameter that strikes a balance between the two types of losses. The overall optimization problem in this study can be expressed as:

$$G^*_{\mathcal{X}}, G^*_{\mathcal{Y}} = \arg\max_{G_{\mathcal{X}}, G_{\mathcal{Y}}} \min_{F_{\mathcal{M}}, D_{\mathcal{X}}, D_{\mathcal{Y}}} \mathcal{L}(G_{\mathcal{X}}, G_{\mathcal{Y}}, F_{\mathcal{M}}, D_{\mathcal{X}}, D_{\mathcal{Y}}). \quad [1]$$

After an iterative model training process, we obtain $(\widehat{G}_{\mathcal{X}}, \widehat{G}_{\mathcal{Y}})$ as estimators for $(G^*_{\mathcal{X}}, G^*_{\mathcal{Y}})$. Moreover, by controlling the distribution on $\mathcal{X}$, we can achieve the following objectives: 1) Manifold estimating: $\widehat{G}_{\mathcal{X}}$ and $\widehat{G}_{\mathcal{Y}}$ estimate the Riemannian exponential and logarithmic maps correspondingly. The image set $\widehat{\mathcal{M}} = \widehat{G}_{\mathcal{X}}(\mathcal{X})$ serves as an estimator for $\mathcal{M}$, with the smoothness and dimensionality of $\widehat{\mathcal{M}}$ each easily controllable. 2) Noise cancelling: the composite mapping $\widehat{G}_{\mathcal{X}} \circ \widehat{G}_{\mathcal{Y}}$ projects any observed point y_i to $\widehat{y}_i \in \widehat{\mathcal{M}}$, which is less affected by noise. 3) Nonlinear interpolating: for points y_i and y_j, we can study the curves between them by applying $\widehat{G}_{\mathcal{X}}$ to interior point between $\widehat{G}_{\mathcal{Y}}(y_i)$ and $\widehat{G}_{\mathcal{Y}}(y_j)$.

Implementation

Network architecture. Within our framework, individual sub-networks are adaptable to cater to specific needs, provided the input and output dimensions resonate with the data. Practically, we can tailor the model to a size that, over extended epochs, may lead to overfitting. Subsequently, we select optimal model weights for inference, drawing from the loss functions observed during training. A concise elaboration on this is available in the SI Appendix supporting text and Fig. S2 and Fig. S3.

To accommodate the intricate nature of subsequent simulation experiments, we have devised several pre-defined models. These models are available in our GitHub repository (https://github.com/zhigang-yao/MFCGAN). In the vector space cases, both the generators and discriminators consist of fully connected neural networks. These networks feature hidden layers with a width of 100 and a depth of 9. Conversely, for data represented in image space, we employ a fully connected neural network as the generator, incorporating hidden layers of 400 in width and 15 in depth. As for the discriminator, we employ a 3-layer PatchGAN discriminator (29) with a kernel size of 4×4. All fully connected neural networks utilize the rectified linear unit (ReLU, (30)) activation function between linear layers, without incorporating Dropout.

To facilitate the sub-module $F_{\mathcal{M}}$, we adhere to the simplified algorithm outlined in (1). For a more complete formulation, please refer to the SI Appendix. After initializing the manifold fitting sub-module with the sample set $\mathcal{Y}_N$ and three radii, namely (r_0, r_1, r_2), we proceed with the following steps for a given input data point y. Initially, we compute the weighted average of y with respect to $\mathcal{Y}_N$ within a radius of r_0, denoting it as μ_y. Subsequently, after leveraging the directionality of $y - \mu_y$, we construct a hypercylinder with radii r_1 and r_2. The weighted average within this hypercylinder serves as the output of the manifold fitting sub-module, namely $F_{\mathcal{M}}(y)$. By shifting points in close proximity to the manifold towards the manifold itself, this sub-module substantially boosts the discriminator's ability to discern crucial distinctions, thereby expediting the training process and ameliorating the overall performance of the generator.

Training details. To address the optimization problem outlined in Eq. (1), we employ the Adam optimizer (31). Initially, the optimizer trains using a predetermined learning rate, which subsequently decreases in a linear fashion after a specified number of epochs. In all subsequent investigations, the distribution of $X \in \mathcal{X}$ is consistently set as a d-dimensional uniform distribution on $(0, 1)^d$.

To further enhance the training effectiveness of the generator and achieve the desired manifold fitting objective, we use two supplementary techniques. Firstly, we dissociate the noise component from $G_{\mathcal{X}}$, creating 30 replicated instances from each $G_{\mathcal{X}}(x)$ during each forward step and incorporating iid noise into them. By adjusting the associated loss function, we amplify the representational capacity of $G_{\mathcal{X}}$ with regard to the manifold. Secondly, within each optimization iteration, we initially hold $D_{\mathcal{X}}$, $D_{\mathcal{Y}}$, and $F_{\mathcal{M}}$ fixed and perform 20 optimization steps for $G_{\mathcal{X}}$ and $G_{\mathcal{Y}}$. We then retain the learned parameters of $G_{\mathcal{X}}$ and $G_{\mathcal{Y}}$ while executing a single optimization step for $D_{\mathcal{X}}$, $D_{\mathcal{Y}}$, and $F_{\mathcal{M}}$. This procedural approach produces theoretical advantages by facilitating

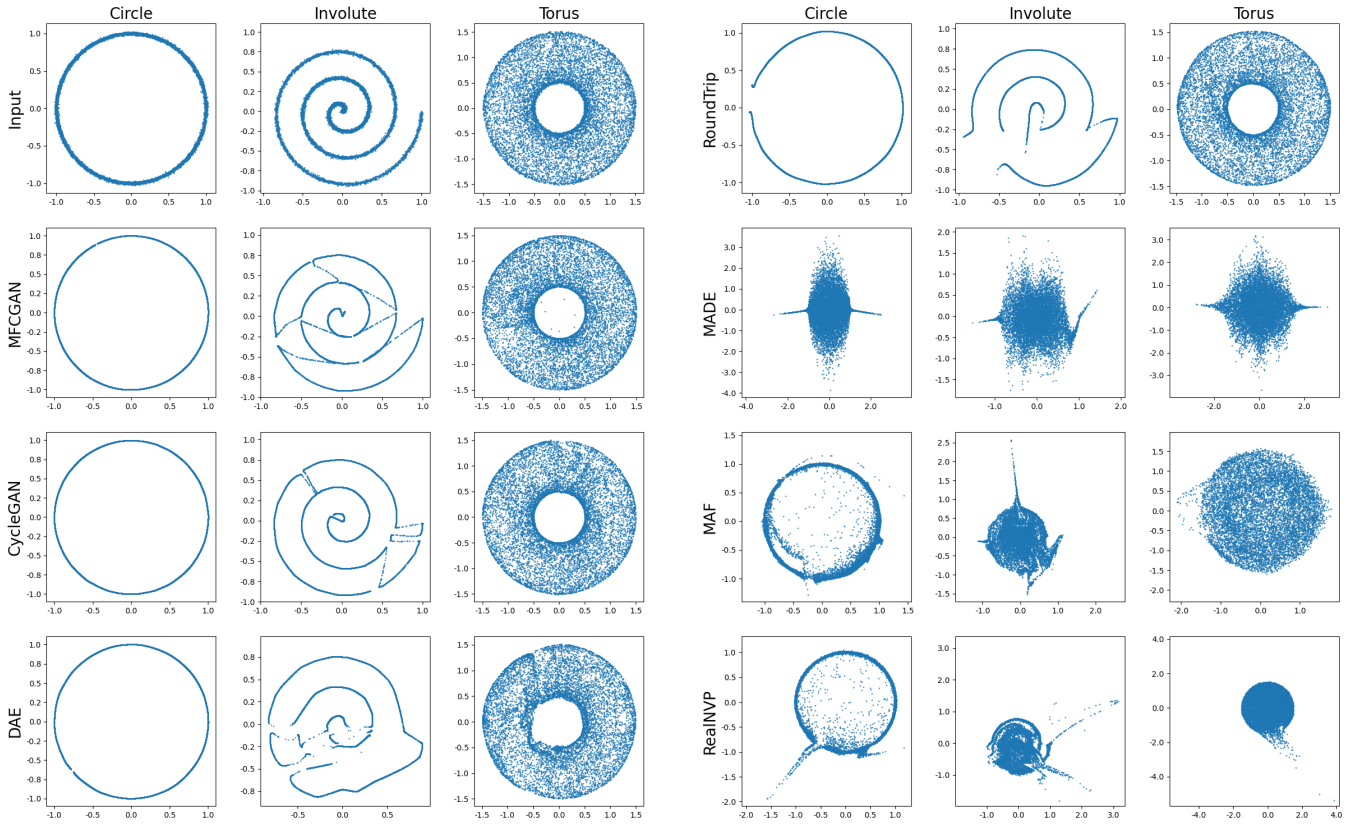


Fig. 2. Comparative scatter plots in Euclidean space for three simulation cases. Each plot presents the input training set or newly generated sample sets from MFCGAN, CycleGAN, RoundTrip, MADE, MAF, and RealNVP, alongside denoised data sets obtained through DAE.

the training process of $G_{\mathcal{X}}$ and $G_{\mathcal{Y}}$, ultimately leading to improved performance.

Results

We initiate our analysis with a comparative assessment of our manifold fitting method, termed `MFCGAN`, juxtaposed against contemporary neural network-based techniques and established manifold fitting methods across several simulated vector-valued data sets. Subsequently, we underscore the adaptability and relevance of our algorithm by highlighting its performance across diverse image space domains. Owing to space constraints, results regarding different σ and additional figures are detailed in the SI Appendix Figs. S4 to S6.

Simulation in Euclidean spaces. To enable a thorough comparison of different models in Euclidean space, we have devised three distinct data sets that exhibit diverse geometries and dimensionalities.

- Circle: Sample $\alpha_i \overset{\text{iid}}{\sim} Unif(0,1)$ and $\mathbf{z}_i \overset{\text{iid}}{\sim} \mathcal{N}(0, I_2)$ for $i = 1, \cdots, N$, where $\mathcal{N}(0, I_2)$ denotes the 2-dimensional standard Gaussian distribution. The sample points are calculated as $\{(\cos(2\pi\alpha_i),\ \sin(2\pi\alpha_i)) + 0.01\mathbf{z}_i \in \mathbb{R}^2\}_{i=1}^N$.

- Involute: Sample $\alpha_i \overset{\text{iid}}{\sim} Unif(0,1)$ and $\mathbf{z}_i \overset{\text{iid}}{\sim} \mathcal{N}(0, I_2)$ for $i = 1, \cdots, N$. The sample points are calculated as $\{(\alpha_i \cos(6\pi\alpha_i),\ \alpha_i \sin(6\pi\alpha_i)) + 0.01\mathbf{z}_i \in \mathbb{R}^2\}_{i=1}^N$.

- Torus: Sample $\alpha_i \overset{\text{iid}}{\sim} Unif(0,1)$, $\beta_i \overset{\text{iid}}{\sim} Unif(0,1)$, and $\mathbf{z}_i \overset{\text{iid}}{\sim} \mathcal{N}(0, I_3)$ for $i = 1, \cdots, N$. Then, the data set can be expressed as $\{(y_{i,1}, y_{i,2}, y_{i,3}) + 0.01\mathbf{z}_i \in \mathbb{R}^3\}_{i=1}^N$, with

$$\begin{cases} y_{i,1} = (1 + \cos(2\pi\alpha_i)/2)\cos(2\pi\beta_i) \\ y_{i,2} = (1 + \cos(2\pi\alpha_i)/2)\sin(2\pi\beta_i) \\ y_{i,3} = \sin(2\pi\alpha_i)/2. \end{cases}$$

By following these steps, the sample set $\mathcal{Y}_N$ can be obtained. Throughout all subsequent experiments, we maintain a consistent sample size of $N = 10^4$. Furthermore, to meet the criteria of MFCGAN, we draw additional 10^4 data points, uniformly distributed in $(0, 1)$ across their respective dimensions, serving as samples from $\mathcal{X}$.

Comparison with neural network-based methods. To conduct a thorough comparison, we implement and train several alternative neural network-based methods, including CycleGAN (26), denoising autoencoder (DAE, (32)), RoundTrip (27), Masked Autoencoder for Distribution Estimation (MADE, (33)), Masked Autoregressive Flow (MAF, (34)), and real-valued non-volume preserving transformation method (RealNVP, (35)). We implement the CycleGAN and DAE methods independently, ensuring they share the same network architecture, hyperparameters, and tuning parameters as the proposed MFCGAN model. We utilize version 2.0.1 of RoundTrip, available on the Python Package Index (PyPI). During the training step, we adjust only the dimensions of the

Table 1. Summary of the distance from the sample points to the latent manifold with respect to seven methods in three simulated cases.

		Circle	Involute	Torus
Input	Mean (SD)	8.13×10^{-3} (6.14×10^{-3})	8.03×10^{-3} (6.04×10^{-3})	7.98×10^{-3} (6.04×10^{-3})
	95% quantile	0.02	0.02	0.02
	Percent $> 3\sigma$	0.42%	0.33%	0.22%
MFCGAN	Mean (SD)	5.66×10^{-4} (4.44×10^{-4})	5.12×10^{-3} (0.02)	2.57×10^{-3} (9.10×10^{-3})
	95% quantile	1.60×10^{-3}	0.01	5.51×10^{-3}
	Percent $> 3\sigma$	0.00%	3.51%	0.30%
CycleGAN	Mean (SD)	1.63×10^{-3} (1.08×10^{-3})	5.75×10^{-3} (0.01)	3.79×10^{-3} (0.01)
	95% quantile	3.65×10^{-3}	0.01	7.66×10^{-3}
	Percent $> 3\sigma$	0.00%	2.32%	1.06%
DAE	Mean (SD)	1.31×10^{-3} (9.09×10^{-4})	0.03 (0.05)	0.02 (0.03)
	95% quantile	3.04×10^{-3}	0.15	0.07
	Percent $> 3\sigma$	0.00%	23.27%	23.90%
RoundTrip	Mean (SD)	0.02 (8.06×10^{-3})	0.02 (0.02)	0.02 (0.01)
	95% quantile	0.03	0.08	0.04
	Percent $> 3\sigma$	12.10%	15.30%	13.14%
MADE	Mean (SD)	0.37 (0.30)	0.09 (0.08)	0.19 (0.18)
	95% quantile	0.88	0.17	0.48
	Percent $> 3\sigma$	93.57%	79.86%	87.75%
MAF	Mean (SD)	0.03 (0.06)	0.08 (0.09)	0.09 (0.10)
	95% quantile	0.13	0.16	0.32
	Percent $> 3\sigma$	23.55%	71.16%	59.21%
RealNVP	Mean (SD)	0.03 (0.09)	0.07 (0.14)	0.08 (0.14)
	95% quantile	0.14	0.15	0.32
	Percent $> 3\sigma$	13.45%	54.67%	46.87%

domains and set the number of iterations to 10^6 in the default parameter list. For the other methods based on normalizing flows, we adopt an implementation obtained from GitHub (downloaded as committed on Jan 22, 2020) at https://github.com/kamenbliznashki/normalizing_flows/tree/master. During training, we set the number of epochs to 200 and enable the option of `no_batch_norm`. Since these methods are designed initially for density estimation, we make necessary code modifications to allow for saving of generated samples from the estimated distributions for all four methods, ensuring compatibility with our evaluation framework.

Following extensive training on data sets consisting of 10^4 samples, we employ each of the aforementioned models, including MFCGAN, to generate additional 10^4 sample points. For both MFCGAN and CycleGAN, we employ uniformly generated random samples as inputs to their respective generators, that is, $G_{\mathcal{X}}$'s. In the case of DAE, we utilize its encoder-decoder architecture to generate denoised data points. With RoundTrip, we leverage its integrated sampler to supply inputs to its generator. For the other methods under consideration, we retain the intermediate samples produced during the estimation phase for subsequent comparison.

To provide a clear visual representation of the generated samples from each algorithm, we present scatter plots of these samples in Fig. 2, where only the 2D projection is plotted for the 3D torus case. It is evident that the points generated by MFCGAN, exhibiting a smooth distribution, closely fit the underlying latent manifold. This outcome underscores the effectiveness of our method in achieving manifold fitting objectives, wherein the resulting estimator manifests as a smooth manifold, particularly in these simulated scenarios. In comparison to the other methods, MFCGAN exhibits the ability to capture and identify more intricate details of the manifold within the generated distribution. This observation suggests that MFCGAN is a promising approach for applications in more complex tasks.

In the case of the involute data set (columns 2 and 5 in Fig. 2), MFCGAN and CycleGAN generate numerous additional lines. The extra lines can be attributed to the fact that the latent involute manifold possesses two boundary points, and that our approach tends to maintain closed image sets for $G_{\mathcal{X}}$. Consequently, the optimization process sacrifices certain low-probability regions (the extra lines) to reconnect the head and tail of the output, thus preserving the closed structure.

To provide a quantitative assessment of the output generated by each method, we calculate the distances from sample points to the latent manifold and compile the results in Tab. 1. The table includes the results of all seven methods, including the training data for reference. We employed three primary metrics: the mean and 95% percentile distance, and the percentage of samples beyond 3σ.

Upon inspection of the table, it is evident that the output of MFCGAN exhibits closer proximity to the latent manifold compared to the training data. This finding once again highlights the ability of MFCGAN to obtain accurate estimates of the manifold structure. Moreover, for manifolds with fixed curvature, the estimation error from MFCGAN may be a higher-order term of σ, although further investigation is necessary to validate this claim due to the absence of a suitable metric.

Notably, the results of MFCGAN significantly outperform the other methods in terms of proximity to the latent manifold. This disparity is reasonable given that density estimation is the primary objective of the other methods. However, it is

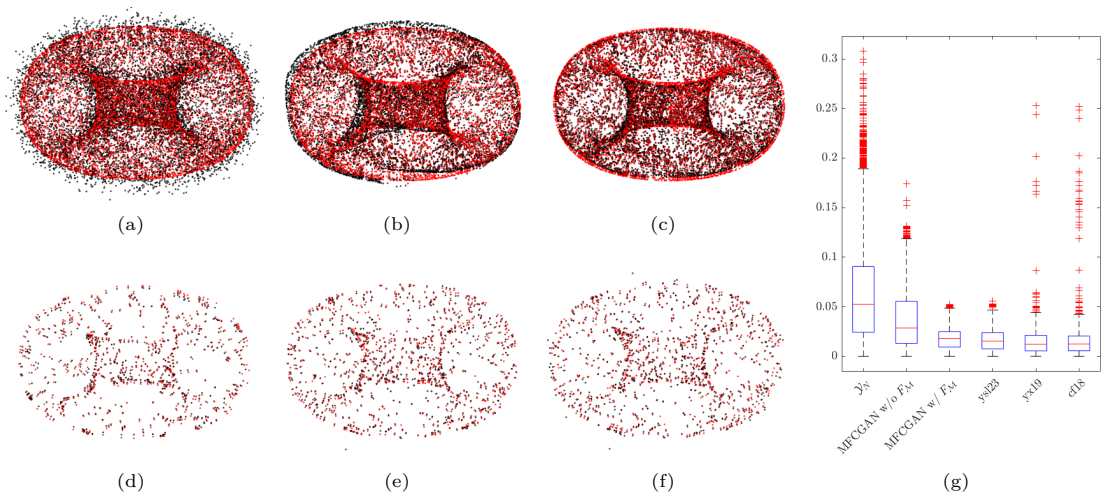


Fig. 3. Comparison of MFCGAN with other manifold fitting algorithms in the torus case. Black points represent (a) the input data $\mathcal{Y}_N$ with noise level $\sigma = 0.08$ and the projections of $\mathcal{Y}_N$ onto the manifolds fitted by (b) MFCGAN w/o $F_{\mathcal{M}}$, (c) MFCGAN w/ $F_{\mathcal{M}}$, (d) ysl23, (e) yx19, and (f) cf18. Red points indicate the projections of the black points onto the latent manifold. (g) A boxplot showcasing the distances between corresponding red and black points.

worth noting that MFCGAN does lag the other methods in terms of the tail distribution, which is a meaningful indicator for assessing the ability of MFCGAN to fit well with high probability. This aspect warrants further investigation in subsequent research endeavors.

Comparison with established manifold fitting methods. Numerous algorithms for manifold fitting have been proposed in the literature, as highlighted by works such as ysl23 (1), yx19 (20), and cf18 (19). To benchmark our method against these, we conduct simulations across the three distinct scenarios using the algorithms hosted at https://github.com/zhigang-yao/manifold-fitting. Fig. 3 showcases the results of the torus case, with additional outcomes of the other two simulation cases available in the SI Appendix. The parameters for these simulations align with those detailed in the preceding subsection. For MFCGAN, all samples in $\mathcal{Y}_N$ are mapped onto the identified manifold with the transformations $\widehat{G}_{\mathcal{X}} \circ \widehat{G}_{\mathcal{Y}}$ (MFCGAN w/o $F_{\mathcal{M}}$) and $F_{\mathcal{M}} \circ \widehat{G}_{\mathcal{X}} \circ \widehat{G}_{\mathcal{Y}}$ (MFCGAN w/ $F_{\mathcal{M}}$). Conversely, traditional manifold fitting methods project a subset of samples onto their respective fitted manifolds, a choice driven by computational considerations.

In terms of fitting error, the traditional manifold fitting methods exhibit a marginally better performance than MFCGAN, an outcome that aligns with theoretical expectations. Given that these methods directly measure the manifold using the Hausdorff distance, a strong alignment of results is expected. MFCGAN, on the other hand, seeks to approximate the distribution of $\omega \star \phi_\sigma$ with $G_{\mathcal{X}}(X) \star \phi_\sigma$, based on specific divergence metrics. However, traditional approaches necessitate that query data remain proximate to the latent manifold, a constraint that can result in isolated points, as evident in panels (e) to (g) of Fig. 3. Such a limitation does not constrain MFCGAN.

From a computational vantage point, MFCGAN notably excels in its capacity to efficiently generate new data on the fitted manifold and project data points onto it, even when accounting for its training duration. In contrast, traditional manifold fitting methods, such as yx19 and cf18, demand rigorous iterative computations for point projection, with their computational requirements escalating as the dimensionality of the ambient space increases. Notably, in our simulation cases, yx19 and cf18 consume several minutes, sometimes extending to ten minutes, to process merely 1,000 points. While the streamlined algorithm of ysl23 operates swiftly, it lacks assurances regarding dimensionality. In stark contrast, MFCGAN completes the projection, with exactly dimension d, for 10,000 points in a mere ten seconds. This efficiency stems from MFCGAN's innovative strategy of harnessing two generative networks to emulate exponential and logarithmic maps, positioning it as a potentially more adept solution for high-dimensional data scenarios.

In summary, while traditional manifold fitting methods have an edge in fitting accuracy, MFCGAN emerges as a compelling contender, especially when considering high-dimensional data and computational efficiency. The observed differences in fitting error between the two methodologies highlight avenues for future exploration, especially in the development of improved discriminators to refine the fitting error assessment of MFCGAN.

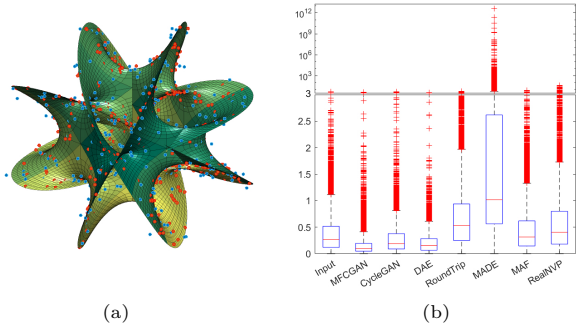
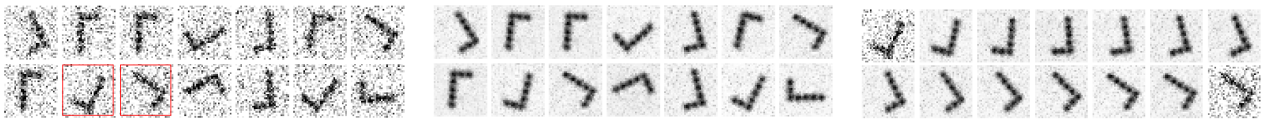


Fig. 4. Fitting a Calabi-Yau manifold. (a) 3D projection of the manifold (surface) with samples from noisy inputs (blue points) and their projection with MFCGAN (red points). (b) boxplot showing the pseudo-bias of input data and outputs from neural network-based methods; the upper part is presented in log scale.



(a) (b) (c)

Fig. 5. Results of MFCGAN on image data. (a) A group of random samples from the training set; (b) The noise reduction result from MFCGAN, obtained by passing images in (a) to $\widehat{G}_{\mathcal{X}} \circ \widehat{G}_{\mathcal{Y}}$; (c) Nonlinear interpolation of the two boxed images in (a), calculated by feeding linear combinations of $\widehat{G}_{\mathcal{Y}}(y_i)$ and $\widehat{G}_{\mathcal{Y}}(y_j)$ to $\widehat{G}_{\mathcal{X}}$.

Simulation with Calabi-Yau manifolds. Calabi–Yau manifolds, as described by Calabi (36), represent a distinguished class of compact, complex Kähler manifolds characterized by a vanishing first Chern class. Their zero Ricci curvature renders them congruent with the theoretical models of the universe postulated by physicists. In this subsection, we explore a specific instance of Calabi-Yau manifolds: a reduced Fermat quartic, whose point $y = (y_1, y_2)$ satisfies

$$y_1^4 + y_2^4 = 1, \quad y \in \mathbb{C}^2. \qquad [2]$$

To construct the training set, we commence by creating a uniform mesh grid comprising $N = 313,296$ points on the manifold specified by Eq. (2). Subsequently, we perturb both the real and imaginary components of each point by adding Gaussian noise with a standard deviation of $\sigma = 0.06$. This process yields the data set $\mathcal{Y}_N$. Fig. 4(a) depicts a 3D projection of the Fermat quartic alongside some points from $\mathcal{Y}_N$. Additional information about data generation and projection is available in the SI Appendix.

Given the computational challenges posed by the sample size, our comparison is limited to the neural network-based approaches. The model structure and the majority of hyperparameters remain consistent with those used in the previous subsection. However, there are adjustments in the training regimen for MFCGAN, CycleGAN, and DAE. Specifically, these models are trained with a constant learning rate for an initial 20 epochs, followed by an additional 40 epochs with a decreasing learning rate.

To evaluate the learning results, a subset of $\mathcal{Y}_N$ with size 10^4 is projected onto the learned manifolds of MFCGAN, CycleGAN, and DAE. For the remaining density estimation methods, additional 10^4 samples are generated with the same methods described before. Due to the complexity of calculating the distance from a point to the manifold, we use a pseudo-bias defined as the norm of $y_1^4 + y_2^4 - 1$ for $y \in \mathbb{C}^2$.

The pseudo-bias of input data and outputs from various methods are encapsulated in the boxplot shown in Fig. 4(b). The primary segment of the box corresponding to MFCGAN exhibits the narrowest distribution range, indicating that the projections of sample points on the manifolds learned by MFCGAN have reduced pseudo-bias. This suggests that MFCGAN fits the latent Calabi-Yau manifold more accurately than other methods, implying its potential applicability to explore more intricate geometric structures.

Application in images. To provide examples of the potential applications of our approach, we conduct a series of experiments on image data sets. We initiate the process by constructing a basic graph with the density function of a Gaussian mixture model, with the component means resembling an "L" shape and subsequently rotating the components in the plane by 10°. A total of 36 resulting images are captured, which are replicated 15 times and white noise is added. Then, we compile a data set comprising 540 images with size 32×32 in pixels (as illustrated in Fig. 5(a)). In theory, these images are positioned around a 1-manifold, yet they are embedded in $\mathbb{R}^{32\times32}$. Our objective is to use MFCGAN to learn the features of this particular embedding.

One potential application of MFCGAN is to remove noise from signals. As depicted in Fig. 5, we utilize the noisy images shown in Fig. 5(a) as input and pass them through the function $\widehat{G}_{\mathcal{X}} \circ \widehat{G}_{\mathcal{Y}}$ to obtain cleaner images, as shown in Fig. 5(b). Remarkably, the orientation of the graph in the output images remains largely consistent with that of the original, which would indicate that the cycle consistency loss component effectively contributes to the training process. In other words, we can employ $\widehat{G}_{\mathcal{X}} \circ \widehat{G}_{\mathcal{Y}}$ as a projection function to map points in $\mathcal{Y}$ onto the latent manifold, thereby achieving an objective that is similar to noise reduction.

Another valuable application of MFCGAN is nonlinear interpolation. We select two graphs from Fig. 5(a), labeled as y_i and y_j. By computing $\widehat{G}_{\mathcal{X}}(t\widehat{G}_{\mathcal{Y}}(y_i) + (1-t)\widehat{G}_{\mathcal{Y}}(y_j))$ for various values of t, we can generate a series of images that lie between y_i and y_j, as depicted in Fig. 5(c). Notably, these interpolated images are not present in the training set, yet they effectively capture the geometric connection between y_i and y_j, resulting in a smooth transition from y_i to y_j. This outcome demonstrates that MFCGAN enables us not only to project points onto the manifold but also to construct a neighborhood of y on the manifold for any given point y by leveraging the information obtained from $G_{\mathcal{Y}}(y)$. Such capabilities are of great importance for further statistical studies on non-Euclidean data.

We also conducted experiments on two sets of real portrait photographs. However, only one result is included here due to space limitations. Further details can be found in the SI Appendix. For our task, we find a black and white GIF featuring Charlie Chaplin, from which 34 frames of 160×160 black and white bitmaps can be extracted. The training set consists of the odd-numbered frames, which undergo data augmentation through the addition of white noise 20 times per frame, while the even-numbered frames remain unchanged and serve as the test set. The neural network structures are consistent with the description provided in the previous section. Following 200 epochs of training the entire network, we evaluate our method by inputting the test set into $\widehat{G}_{\mathcal{X}} \circ \widehat{G}_{\mathcal{Y}}$ to assess its ability to capture manifold information not present in the training set.

Fig. 6 presents an intriguing result, showcasing frames 9 and 11 of the GIF on the left- and right-most panels, respectively, with frame 10 and its recovery depicted in the two center panels. Notably, the recovered image is more

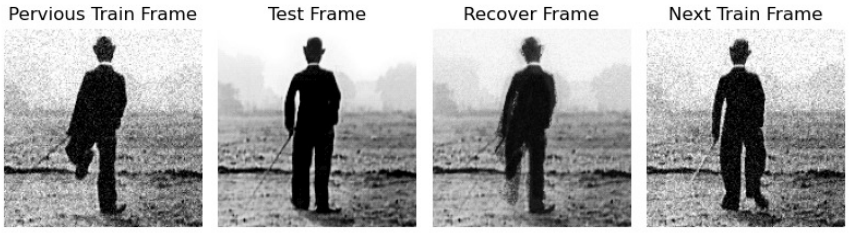


Fig. 6. Results on real data: Projection of test image onto the learned manifold and comparison with the previous and next training frame.

than just a simple average of the previous and next frames, as evident from the angle of the cane and the shape of the legs. This suggests that MFCGAN effectively captures implicit information closer to frame 10 that is absent from the training data. However, the clarity of the recovered data is compromised due to the absence of image order labels and the limited data size during training. The model may sort the images in a different order than their temporal order. Additional details can be found in the SI Appendix.

Discussion

In this study, we introduce MFCGAN as a novel approach for manifold fitting, utilizing a modified framework derived from CycleGAN. In accordance with the existing theoretical framework, we address several modeling assumptions in our work. However, it is essential to reiterate that the assumptions related to uniform and Gaussian distributions, as well as the knowledge of d and σ, are subject to potential relaxation. Our proposed method effectively captures embedded manifold information in high-dimensional data, as demonstrated by extensive numerical simulations and real data experiments. Nonetheless, some areas require improvement. Specifically, the current algorithm demands considerable storage space while enhancing training efficiency is essential too. Furthermore, future research should focus on extending our computational approach to incorporate theoretical advancements in manifold fitting.

Data Archival. The PyTorch-based implementation, encompassing generators, discriminators, and manifold fitting submodule of the model, along with the requisite code for data generation and comparison in simulations, is accessible via https://github.com/zhigang-yao/MFCGAN.

ACKNOWLEDGMENTS. Z.Y. has been supported by MOE Tier 2 grant (A-0008520-00-00, A-8001562-00-00) and Tier 1 grant (A-0004809-00-00, A8000987-00-00) at the National University of Singapore; J.S. is supported by Research Assistantship under grant A-0004826-00-00 at the National University of Singapore. Part of the work comes from Z.Y.'s collaboration with S.Y. at the Center of Mathematical Sciences And Applications (CMSA) at Harvard University.

1. Z Yao, J Su, B Li, Manifold fitting: An invitation to statistics. *arXiv preprint arXiv:2304.07680* (2023).
2. JB Tenenbaum, Vd Silva, JC Langford, A global geometric framework for nonlinear dimensionality reduction. *science* **290**, 2319–2323 (2000).
3. ST Roweis, LK Saul, Nonlinear dimensionality reduction by locally linear embedding. *science* **290**, 2323–2326 (2000).
4. DL Donoho, C Grimes, Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proc. Natl. Acad. Sci.* **100**, 5591–5596 (2003).
5. M Belkin, P Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation* **15**, 1373–1396 (2003).
6. Z Zhang, H Zha, Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *SIAM journal on scientific computing* **26**, 313–338 (2004).
7. L McInnes, J Healy, J Melville, Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426* (2018).
8. S Deutsch, A Ortega, G Medioni, Manifold denoising based on spectral graph wavelets in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. (IEEE), pp. 4673–4677 (2016).
9. T Yang, J Meng, Manifold fitting algorithm of noisy manifold data based on variable-scale spectral graph. *Soft Comput.* pp. 1–12 (2021).
10. S Luo, W Hu, Differentiable manifold reconstruction for point cloud denoising in *Proceedings of the 28th ACM international conference on multimedia*. pp. 1330–1338 (2020).
11. W Wang, MA Carreira-Perpinán, Manifold blurring mean shift algorithms for manifold denoising in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. (IEEE), pp. 1759–1766 (2010).
12. B Sober, D Levin, Manifold approximation by moving least-squares projection (mmls). *Constr. Approx.* **52**, 433–478 (2020).
13. C Genovese, M Perone-Pacifico, I Verdinelli, L Wasserman, Minimax manifold estimation. *J. Mach. Learn. Res.* **13**, 1263–1291 (2012).
14. CR Genovese, M Perone-Pacifico, I Verdinelli, L Wasserman, Manifold estimation and singular deconvolution under hausdorff loss. *The Annals Stat.* **40**, 941–963 (2012).
15. CR Genovese, M Perone-Pacifico, I Verdinelli, L Wasserman, Nonparametric ridge estimation. *The Annals Stat.* **42**, 1511–1545 (2014).
16. YC Chen, CR Genovese, L Wasserman, Asymptotic theory for density ridges. *The Annals Stat.* **43**, 1896–1928 (2015).
17. K Mohammed, H Narayanan, Manifold learning using kernel density estimation and local principal components analysis. *arXiv preprint arXiv:1709.03615* (2017).
18. DB Dunson, HT Wu, N Wu, Graph based gaussian processes on restricted domains. *J. Royal Stat. Soc. Ser. B: Stat. Methodol.* **84**, 414–439 (2022).
19. C Fefferman, S Ivanov, Y Kurylev, M Lassas, H Narayanan, Fitting a putative manifold to noisy data in *Conference On Learning Theory*. (PMLR), pp. 688–720 (2018).
20. Z Yao, Y Xia, Manifold fitting under unbounded noise. *arXiv preprint arXiv:1909.10228* (2019).
21. C Fefferman, S Ivanov, M Lassas, H Narayanan, Fitting a manifold of large reach to noisy data. *arXiv preprint arXiv:1910.05084* (2021).
22. Z Yao, Y Xia, Z Fan, Random fixed boundary flows. *J. Am. Stat. Assoc.* **0**, 1–22 (2023).
23. VM Panaretos, T Pham, Z Yao, Principal flows. *J. Am. Stat. Assoc.* **109**, 424–436 (2014).
24. DE Rumelhart, GE Hinton, RJ Williams, , et al., Learning internal representations by error propagation (1985).
25. I Goodfellow, et al., Generative adversarial networks. *Commun. ACM* **63**, 139–144 (2020).
26. JY Zhu, T Park, P Isola, AA Efros, Unpaired image-to-image translation using cycle-consistent adversarial networks in *Proceedings of the IEEE international conference on computer vision*. pp. 2223–2232 (2017).
27. Q Liu, J Xu, R Jiang, WH Wong, Density estimation using deep generative neural networks. *Proc. Natl. Acad. Sci.* **118**, e2101344118 (2021).
28. X Mao, et al., Least squares generative adversarial networks in *Proceedings of the IEEE international conference on computer vision*. pp. 2794–2802 (2017).
29. P Isola, JY Zhu, T Zhou, AA Efros, Image-to-image translation with conditional adversarial networks in *Proceedings of the IEEE conference on computer vision and pattern recognition*. pp. 1125–1134 (2017).
30. AF Agarap, Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375* (2018).
31. DP Kingma, J Ba, Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
32. P Vincent, H Larochelle, Y Bengio, PA Manzagol, Extracting and composing robust features with denoising autoencoders in *Proceedings of the 25th international conference on Machine learning*. pp. 1096–1103 (2008).
33. M Germain, K Gregor, I Murray, H Larochelle, Made: Masked autoencoder for distribution estimation in *International conference on machine learning*. (PMLR), pp. 881–889 (2015).
34. G Papamakarios, T Pavlakou, I Murray, Masked autoregressive flow for density estimation. *Adv. neural information processing systems* **30** (2017).
35. L Dinh, J Sohl-Dickstein, S Bengio, Density estimation using real nvp. *arXiv preprint arXiv:1605.08803* (2016).
36. E Calabi, On kähler manifolds with vanishing canonical class in *Algebraic geometry and topology. A symposium in honor of S. Lefschetz*. Vol. 12, pp. 78–89 (2015).