2 Related Work

The geometric perspective on data is ubiquitous in machine learning. Geometric techniques enhance unsupervised and semi-supervised learning, generative and discriminative models [9; 5; 37]. We outline the applications of the proposed manifold comparison technique and highlight the geometric intuition along the way.

2.1 Generative Model Evaluation

Past research has explored many different directions for the evaluation of generative models. Setting aside models that ignore the true data distribution, such as the Inception Score [45] and GILBO [3], we discuss most relevant geometric ideas below; we refer the reader to [12] for a comprehensive survey.

Critic model-based metrics. Classifier two-sample tests (C2ST) [36] aim to assess whether two samples came from the same distribution by means of an auxiliary classifier. This idea is reminiscent of the GAN discriminator network [23]: if it is possible to train a model that distinguishes between samples from the model and the data distributions, it follows that these distributions are not entirely similar. The convergence process of the GAN-like discriminator [5; 11] lends itself to creating a family of metrics based on training a discriminative classifier [30]. Still, training a separate critic model is often computationally prohibitive and requires careful specification. Besides, if the critic model is a neural network, the resulting metric lacks interpretability and training stability. Human evaluators may act as critic classifiers [17; 55], yet may confer no more interpretability and stability than a neural critic model.

A standard GAN model can be seen as a C2ST for the integral probability metric between the model and the data distributions. Many advanced GAN models such as Wasserstein, MMD, Sobolev and Spherical GANs impose different constraints on the function class so as to stabilize training [5; 11; 39; 42]. Higher-order moment matching [11; 42] enhances GAN performance, enabling GANs to capture multi-scale data properties, while multi-scale noise ameliorates GAN convergence problems [31]. Still, no feasible multi-scale GAN evaluation metric has been proposed to date.

Positional distribution comparison. In certain settings, it is acceptable to assign zero probability mass to the real data points [41]. In effect, metrics that estimate a distribution's location and dispersion provide useful input for generative model evaluations. For instance, the Fréchet Inception Distance (FID) [27] computes the Wasserstein-2 (i.e., Fréchet) distance between distributions approximated with Gaussians, using only the estimated mean and covariance matrices; the Kernel Inception Distance (KID) [11] computes a polynomial kernel $\mathrm{k}(x, y) = (\frac{1}{d} x^\top y + 1)^3$ and measures the associated Kernel Maximum Mean Discrepancy (kernel MMD). Unlike FID, KID has an unbiased estimator [24; 11]. However, even while such methods, based on a limited number of moments, may be computationally inexpensive, they only provide a rudimentary characterization of distributions from a geometric viewpoint.

Intrinsic geometric measures. The Geometry Score [32] characterizes distributions in terms of their estimated persistent homology, which roughly corresponds to the number of holes in a manifold. Still, the Geometry Score assesses distributions merely in terms of their *global* geometry. In this work, we aim to provide a *multi-scale* geometric assessment.

2.2 Similarity of Neural Network Representations

Learning how representations evolve during training or across initializations provides a pathway to the interpretability of neural networks [43]. Still, state-of-the-art methods for comparing representations of neural networks [34; 38; 53] consider only linear projections. The intrinsic nature of the metric we propose renders it appropriate for the task of comparing data representations, which can only rely on intrinsic information.

3 Multi-Scale Intrinsic Distance

At the core of deep learning lies the *manifold hypothesis*, which states that high-dimensional data, such as images or text, lie on a low-dimensional manifold [40; 9; 10]. We aim to provide a theoretically motivated comparison of those underlying data manifolds based on rich intrinsic information. Our target measure should have the following properties:

intrinsic – it is invariant to isometric transformations of the manifold, e.g. translations or rotations.

multi-scale – it captures both local and global information.

We expose our method starting out with heat kernels, which admit a notion of manifold metric and can be used to lower-bound the distance between manifolds.

3.1 Heat Kernels on Manifolds and Graphs

Based on the heat equation, the heat kernel captures *all* the information about a manifold's intrinsic geometry [47]. Given the Laplace-Beltrami operator (LBO) $\Delta_{\mathcal{X}}$ on a manifold $\mathcal{X}$, the *heat equation* is $\frac{\partial u}{\partial t} = \Delta_{\mathcal{X}} u$ for $u : \mathbb{R}^+ \times \mathcal{X} \to \mathbb{R}^+$. A smooth function u is a *fundamental solution* of the heat equation at point $x \in \mathcal{X}$ if u satisfies both the heat equation and the Dirac condition $u(t, \cdot) \to \delta(\cdot - x)$ as $t \to 0^+$. The heat kernel $\mathrm{k}_{\mathcal{X}} \colon \mathcal{X} \times \mathcal{X} \times \mathbb{R}^+ \to \mathbb{R}_0^+$ is the unique solution of the heat equation; while heat kernels can be defined on hyperbolic spaces and other exotic geometries, we restrict our exposition to Euclidean spaces $\mathbb{R}^d$, on which the heat kernel is defined as:

$$\mathrm{k}_{\mathbb{R}^d}(x, x', t) = \frac{1}{(4\pi t)^{d/2}} \exp\left(-\frac{\|x - x'\|^2}{4t}\right), \quad \forall x, x' \in \mathbb{R}^d, t \in \mathbb{R}^+ \tag{1}$$

For a compact $\mathcal{X}$ including $\mathbb{R}^d$, the heat kernel admits the expansion $\mathrm{k}_{\mathcal{X}}(x, x', t) = \sum_{i=0}^{\infty} e^{-\lambda_i t} \phi_i(x) \phi_j(x')$, where λ_i and ϕ_i are the i-th eigenvalue and eigenvector of $\Delta_{\mathcal{X}}$. For $t \simeq 0^+$, according to Varadhan's lemma, the heat kernel approximates geodesic distances.

Heat kernels are also defined for graphs in terms of their Laplacian matrices. An undirected graph is a pair $G = (V, E)$, where $V = (v_1, \ldots, v_n), n = |V|$, is the set of vertices and $E \subseteq (V \times V)$ the set of edges. The *adjacency matrix* of G is a $n \times n$ matrix $\mathbf{A}$ having $\mathbf{A}_{ij} = 1$ if $(i, j) \in E$ and $A_{ij} = 0$ otherwise. The *normalized graph Laplacian* is the matrix $\boldsymbol{\mathcal{L}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$, where $\mathbf{D}$ is the diagonal matrix in which entry $\mathbf{D}_{ii}$ holds the degree of node i, i.e, $\mathbf{D}_{ii} = \sum_{j=1}^{n} \mathbf{A}_{ij}$. Since the Laplacian matrix is symmetric, its eigenvectors $\phi_1, \ldots, \phi_n$, are real and orthonormal. Thus, it is factorized as $\boldsymbol{\mathcal{L}} = \boldsymbol{\Phi} \boldsymbol{\Lambda} \boldsymbol{\Phi}^\top$, where $\boldsymbol{\Lambda}$ is a diagonal matrix with the sorted eigenvalues $\lambda_1 \leq \ldots \leq \lambda_n$, and Φ is the orthonormal matrix $\boldsymbol{\Phi} = (\phi_1, \ldots, \phi_n)$ having the eigenvectors of $\boldsymbol{\mathcal{L}}$ as its columns. The heat kernel on a graph is also given by the solution to the heat equation on a graph, which requires an eigendecomposition of its Laplacian: $\mathbf{H}_t = e^{-t\boldsymbol{\mathcal{L}}} = \boldsymbol{\Phi} e^{-t\boldsymbol{\Lambda}} \boldsymbol{\Phi}^\top = \sum_i e^{-t\lambda_i} \phi_i \phi_i^\top$.

A useful invariant of the heat kernel is the *heat kernel trace* $\mathrm{hkt}_{\mathcal{X}} : \mathcal{X} \times \mathbb{R}_0^+ \to \mathbb{R}_0^+$, defined by a diagonal restriction as $\mathrm{hkt}_{\mathcal{X}}(t) = \int_{\mathcal{X}} \mathrm{k}_{\mathcal{X}}(x, x, t) dx = \sum_{i=0}^{\infty} e^{-\lambda_i t}$ or, in the discrete case, $\mathrm{hkt}_{\boldsymbol{\mathcal{L}}}(t) = \mathrm{Tr}(\mathbf{H}_t) = \sum_i e^{-t\lambda_i}$. Heat kernels traces (HKTs) have been successfully applied to the analysis of 3D shapes [47] and graphs [50]. The HKT contains *all* the information in the graph's spectrum, both local and global, as the eigenvalues λ_i can be inferred therefrom. For example, if there are c disconnected components in the graph, then $\lim_{t \to \infty} \mathrm{hkt}_{\boldsymbol{\mathcal{L}}}(t) = c$.

3.2 Convergence to the Laplace-Beltrami Operator

An important property of graph Laplacians is that it is possible to construct a graph among points sampled from a manifold $\mathcal{X}$ such that the spectral properties of its Laplacian resemble those of the Laplace-Beltrami operator on $\mathcal{X}$. Belkin and Niyogi [9] proposed such a construction, the point cloud Laplacian, which is used for dimensionality reduction in a technique called Laplacian eigenmaps. Convergence to the LBO has been proven for various definitions of the graph Laplacian, including the one we use [10; 26; 15; 49]. We recite the convergence results for the point cloud Laplacian from [10]:

Theorem 1 *Let $\lambda_{n,i}^{t_n}$ and $\phi_{n,i}^{t_n}$ be the i^{th} eigenvalue and eigenvector, respectively, of the point cloud Laplacian $\boldsymbol{\mathcal{L}}^{t_n}$; let λ_i and ϕ_i be the i^{th} eigenvalue and eigenvector of the LBO Δ. Then, there exists*

$t_n \to 0$ *such that*

$$\lim_{n\to\infty} \lambda_{n,i}^{t_n} = \lambda_i$$
$$\lim_{n\to\infty} \left\|\phi_{n,i}^{t_n} - \phi_i\right\|_2 = 0$$

Still, the point cloud Laplacian involves the creation of an $\mathcal{O}(n^2)$ matrix; for the sake of scalability, we use the k-nearest-neighbours (kNN) graph by OR-construction (i.e., based on bidirectional kNN relationships among points), whose Laplacian converges to the LBO for data with sufficiently high intrinsic dimension [49]. As for the choice of k, a random geometric kNN graph is connected when $k \geq {}^{\log n}/_{\log 7} \approx 0.5139 \log n$ [8]; $k = 5$ yields connected graphs for all sample sizes we tested.

3.3 Spectral Gromov-Wasserstein Distance

Even while it is a multi-scale metric *on* manifolds, the heat kernel can be spectrally approximated by finite graphs constructed from points sampled from these manifolds. In order to construct a metric *between* manifolds, Mémoli [37] suggests an optimal-transport-theory-based "meta-distance": a spectral definition of the Gromov-Wasserstein distance between Riemannian manifolds based on matching the heat kernels at all scales. The cost of matching a pair of points (x, x') on manifold $\mathcal{M}$ to a pair of points (y, y') on manifold $\mathcal{N}$ at scale t is given by their heat kernels $\mathrm{k}_\mathcal{M}, \mathrm{k}_\mathcal{N}$:

$$\Gamma(x, y, x', y', t) = \left|\mathrm{k}_\mathcal{M}(x, x', t) - \mathrm{k}_\mathcal{N}(y, y', t)\right|.$$

The distance between the manifolds is then defined in terms of the infimal measure coupling

$$d_{\mathrm{GW}}(\mathcal{M}, \mathcal{N}) = \inf_\mu \sup_{t>0} e^{-2(t+t^{-1})} \|\Gamma\|_{L^2(\mu\times\mu)},$$

where the infimum is sought over all measures μ on $\mathcal{M} \times \mathcal{N}$ marginalizing to the standard measures on $\mathcal{M}$ and $\mathcal{N}$. For finite spaces, μ is a doubly-stochastic matrix. This distance is lower-bounded [37] in terms of the respective heat kernel traces as:

$$d_{\mathrm{GW}}(\mathcal{M}, \mathcal{N}) \geq \sup_{t>0} e^{-2(t+t^{-1})} \left|\mathrm{hkt}_\mathcal{M}(t) - \mathrm{hkt}_\mathcal{N}(t)\right|. \tag{2}$$

This lower bound is the scaled L_∞ distance between the *heat trace signatures* $\mathrm{hkt}_\mathcal{M}$ and $\mathrm{hkt}_\mathcal{N}$. The scaling factor $e^{-2(t+t^{-1})}$ favors medium-scale differences, meaning that this lower bound is not sensitive to local perturbations. The maximum of the scaling factor occurs at $t = 1$, and more than $1 - 10^{-8}$ of the function mass lies between $t = 0.1$ and $t = 10$.

3.4 Heat Trace Estimation

Calculating the heat trace signature efficiently and accurately is a challenge on a large graph as it involves computing a trace of a large matrix exponential, i.e. $\mathrm{Tr}(e^{-t\boldsymbol{\mathcal{L}}})$. A naive approach would be to use an eigendecomposition $\exp(-t\boldsymbol{\mathcal{L}}) = \Phi \exp(-t\Lambda)\Phi^\top$, which is infeasible for large n. Recent work [50] suggested using either truncated Taylor expansion or linear interpolation of the interloping eigenvalues, however, both techniques are quite coarse. To combine accuracy and speed, we use the Stochastic Lanczos Quadrature (SLQ) [51; 20]. This method combines the Hutchinson trace estimator [29; 2] and the Lanczos algorithm for eigenvalues. We aim to estimate the trace of a matrix function with a Hutchinson estimator:

$$\mathrm{Tr}(f(\boldsymbol{\mathcal{L}})) = \mathbb{E}_{p(\mathbf{v})}(\mathbf{v}^\top f(\boldsymbol{\mathcal{L}})\mathbf{v}) \approx \frac{n}{n_v} \sum_{i=1}^{n_v} \mathbf{v}_i^\top f(\boldsymbol{\mathcal{L}})\mathbf{v}_i, \tag{3}$$

where the function of interest $f(\cdot) = \exp(\cdot)$ and $\mathbf{v}_i$ are n_v random vectors drawn from a distribution $p(\mathbf{v})$ with zero mean and unit variance. A typical choice for $p(\mathbf{v})$ is Rademacher or a standard normal distribution. In practice, there is little difference, although in theory Rademacher has less variance, but Gaussian requires less random vectors [7].

To estimate the quadratic form $\mathbf{v}_i^\top f(\boldsymbol{\mathcal{L}})\mathbf{v}_i$ in (3), with a symmetric real-valued matrix $\boldsymbol{\mathcal{L}}$ and a smooth function f, we plug the eigendecomposition $\boldsymbol{\mathcal{L}} = \Phi\Lambda\Phi^\top$ and rewrite the outcome as a Riemann-Stieltjes integral and estimate it with the m-point Gauss quadrature rule [21]:

$$\mathbf{v}_i^\top f(\boldsymbol{\mathcal{L}})\mathbf{v}_i = \mathbf{v}_i^\top \Phi f(\Lambda)\Phi^\top \mathbf{v}_i = \sum_{j=1}^{n} f(\lambda_j)\mu_j^2 = \int_a^b f(t)d\mu(t) \approx \sum_{k=1}^{m} \omega_k f(\theta_k), \tag{4}$$

where $\mu_j = [\Phi^\top \mathbf{v}_i]_j$ and $\mu(t)$ is a piecewise constant function defined as follows

$$\mu(t) = \begin{cases} 0, & \text{if } t < a = \lambda_n \\ \sum_{j=1}^{i} \mu_j^2, & \text{if } \lambda_i \leq t < \lambda_{i-1} \\ \sum_{j=1}^{n} \mu_j^2, & \text{if } b = \lambda_1 \leq t \end{cases}$$

and θ_k are the quadrature's nodes and ω_k are the corresponding weights. We obtain ω_k and θ_k with the m-step Lanczos algorithm. Below we cover this procedure succinctly, for details see [20].

Given the symmetric matrix $\boldsymbol{\mathcal{L}}$ and an arbitrary *starting unit-vector* $\mathbf{q}_0$, the m-step Lanczos algorithm computes an $n \times m$ matrix $\mathbf{Q} = [\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_{m-1}]$ with orthogonal columns and an $m \times m$ tridiagonal symmetric matrix $\mathbf{T}$, such that $\mathbf{Q}^\top \boldsymbol{\mathcal{L}} \mathbf{Q} = \mathbf{T}$. The columns of $\mathbf{Q}$ constitute an orthonormal basis for the Krylov subspace $\mathcal{K}$ that spans vectors $\{\mathbf{q}_0, \boldsymbol{\mathcal{L}}\mathbf{q}_0, \ldots, \boldsymbol{\mathcal{L}}^{m-1}\mathbf{q}_0\}$; each $\mathbf{q}_i$ vector is given as a polynomial in $\boldsymbol{\mathcal{L}}$ applied to the initial vector $\mathbf{q}_0$: $\mathbf{q}_i = p_i(\boldsymbol{\mathcal{L}})\mathbf{q}_0$. These Lanczos polynomials are orthogonal with respect to the integral measure $\mu(t)$. As orthogonal polynomials satisfy the three term recurrence relation, we obtain p_{k+1} as a combination of p_k and p_{k-1}. The tridiagonal matrix storing the coefficients of such combinations, called the Jacobi matrix $\mathbf{J}$, is exactly the tridiagonal symmetric matrix $\mathbf{T}$. A classic result tells us that the nodes θ_k and the weights ω_k of the Gauss quadrature are the eigenvalues of $\mathbf{T}$, λ_k, and the squared first components of its normalized eigenvectors, τ_k^2, respectively (see [22; 54; 20]). Thereby, setting $\mathbf{q}_0 = \mathbf{v}_i$, the estimate for the quadratic form becomes:

$$\mathbf{v}_i^\top f(\boldsymbol{\mathcal{L}})\mathbf{v}_i \approx \sum_{k=1}^{m} \tau_k^2 f(\lambda_k), \quad \tau_k = \mathbf{U}_{0,k} = \mathbf{e}_1^\top \mathbf{u}_k, \quad \lambda_k = \Lambda_{k,k} \quad \mathbf{T} = \mathbf{U}\Lambda\mathbf{U}^\top, \tag{5}$$

Applying (5) over n_v random vectors in the Hutchinson trace estimator (3) yields the SLQ estimate:

$$\operatorname{Tr}(f(\boldsymbol{\mathcal{L}})) \approx \frac{n}{n_v} \sum_{i=1}^{n_v} \left(\sum_{k=0}^{m} \left(\tau_k^i\right)^2 f\left(\lambda_k^i\right) \right) = \Gamma. \tag{6}$$

We derive error bounds for the estimator based on the Lanczos approximation of the matrix exponential, and show that even a few Lanczos steps, i.e., $m = 10$, are sufficient for an accurate approximation of the quadratic form. However, the trace estimation error is theoretically dominated by the error of the Hutchinson estimator, e.g. for Gaussian $p(\mathbf{v})$ the bound on the number of samples to guarantee that the probability of the relative error exceeding ϵ is at most δ is $8\epsilon^{-2}\ln(2/\delta)$ [44]. Although, in practice, we observe performance much better than the bound suggests. Hutchinson error implies nearing accuracy roughly 10^{-2} with $n_v \geq 10$k random vectors, however, with as much as $n_v = 100$ the error is already 10^{-3}. Thus, we use default values of $m = 10$ and $n_v = 100$ in all experiments in Section 4. Please see Appendix A for full derivations and figures.

3.5 Putting MSID Together

We employ the heretofore described advances in differential geometry and numerical linear algebra to create MSID (*Multi-Scale Intrinsic Distance*), a fast, intrinsic method to lower-bound the spectral Gromov-Wasserstein distance between manifolds.

Given data samples in $\mathbb{R}^d$, we build a kNN graph G by OR-construction, such that its Laplacian spectrum approximates the one of the Laplace-Beltrami operator of the underlying manifold [49], and then compute $\operatorname{hkt}_G(t) = \sum_i e^{-\lambda_i t} \approx \Gamma$. We compare heat traces in the spirit of Equation (2), i.e., $|\operatorname{hkt}_{G_1}(t) - \operatorname{hkt}_{G_2}(t)|$ for $t \in (0.1, 10)$ sampled from a logarithmically spaced grid.

Constructing exact kNN graphs is an $\mathcal{O}(dn^2)$ operation; however, approximation algorithms take near-linear time $\mathcal{O}(dn^{1+\omega})$ [19; 6]. As we will see, in practice, with approximate kNN graph construction [19], computational time is low while result variance is similar to the exact case.

The m-step Lanczos algorithm on a sparse $n \times n$ kNN Laplacian $\boldsymbol{\mathcal{L}}$ with one starting vector has $\mathcal{O}(knm)$ complexity, where kn is the number of nonzero elements in $\boldsymbol{\mathcal{L}}$. The symmetric tridiagonal matrix eigendecomposition incurs an additional $\mathcal{O}(m \log m)$ [14]. We apply this algorithm over n_v starting vectors, yielding a complexity of $\mathcal{O}(n_v(m \log m + kmn))$, with constant $k = 5$ and $m = 10$ by default. In effect, MSID's time complexity stands between those of two common GAN evaluation methods: KID, which is $\mathcal{O}(dn^2)$ and FID, which is $\mathcal{O}(d^3 + dn)$. The time complexity of Geometry Score is unspecified in [32], yet in Section 4.4 we show that its runtime grows exponentially in sample size.

Table 1: MSID agrees with KID and FID across varying datasets for GAN evaluation.

	MNIST		FashionMNIST		CIFAR10		CelebA	
Metric	WGAN	WGAN-GP	WGAN	WGAN-GP	WGAN	WGAN-GP	WGAN	WGAN-GP
MSID	**57.74** ± 0.47	**10.77** ± 0.42	**118.14** ± 0.52	**13.45** ± 0.54	**18.10** ± 0.36	**10.84** ± 0.42	**10.11** ± 0.33	**2.84** ± 0.31
KID $\times 10^3$	**47.26** ± 0.07	**5.53** ± 0.03	**119.93** ± 0.14	**25.49** ± 0.07	**93.89** ± 0.09	**59.59** ± 0.09	**217.28** ± 0.14	**92.71** ± 0.08
FID	**31.75** ± 0.07	**8.95** ± 0.03	**152.44** ± 0.12	**35.31** ± 0.07	**101.43** ± 0.09	**80.65** ± 0.09	**205.63** ± 0.09	**85.55** ± 0.08

4 Experiments

Here, we study two key applications of MSID, namely the *evaluation of generative models* and the *study of neural network manifolds*. Further, we provide evidence on the stability of MSID with respect to sample size and different feature spaces. Last, we show that MSID also offers interpretability of results. Even while our metric is not restricted to any specific data representation, we use the Inception network [48] to obtain features, so as to provide a common ground for comparing metrics on generative models. We open-source the code of the method online[1].

4.1 Application of MSID to generative model evaluation

First, we evaluate the sensitivity of MSID, FID, and KID to image transformations. We progressively blur images from the CIFAR-10 training set, and measure the distance to the original data manifold, averaging outcomes over 100 subsamples of 10k images each. To enable comparison across methods, we normalize each distance measure such that the distance between CIFAR-10 and MNIST is 1. Figure 2 reports the results at different levels σ of Gaussian blur. We additionally report the normalized distance to the CIFAR-100 training set (dashed lines). FID and KID quickly drift away from the original distribution and match MNIST, a dataset of a completely different nature. Contrariwise, MSID is robust to noise and recognizes the manifold structure, as the relationships between objects remain mostly unaffected. Moreover, with both FID and KID, low noise ($\sigma = 1$) applied to CIFAR-10 suffices to exceed the distance of CIFAR-100, which is similar to CIFAR-10. MSID is much more robust, exceeding that distance only with $\sigma = 2$.

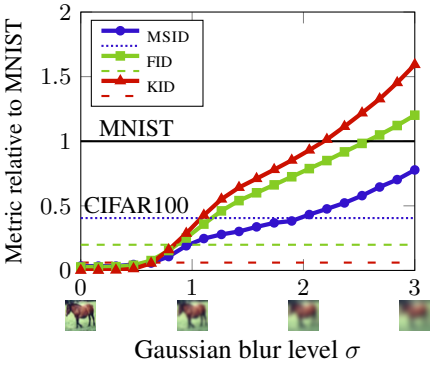

Figure 2: FID, KID and MSID on the CIFAR-10 dataset with Gaussian blur

Next, we turn our attention to the sample-based evaluation of generative models. We train the WGAN [5] and WGAN-GP [25] models on four datasets: MNIST, FashionMNIST, CIFAR10 and CelebA. We sample 10k samples, $\mathbf{Y}$, from each GAN. We then uniformly subsample $10k$ images from the corresponding original dataset, $\mathbf{X}$, and compute the MSID, KID and FID scores between $\mathbf{X}$ and $\mathbf{Y}$. Table 1 reports the average measure and its 99% confidence interval across 100 runs. MSID, as well as both FID and KID, reflect the fact that WGAN-GP is a more expressive model. We provide details on architecture, training, and generated samples in Appendix C.

4.2 Application of MSID to studying neural network manifolds

Next, we employ MSID to inspect the internal dynamics of neural networks.

First, we investigate the stability of output layer manifolds across random initializations. We train 10 instances of the VGG-16 [46] network using different weight initializations on the CIFAR-10 and CIFAR-100 datasets. We compare the average MSID scores across representations in each network layer relative to the last layer. As Figure 3 (left) shows, for both CIFAR-10 and CIFAR-100,

[1] https://github.com/xgfs/msid

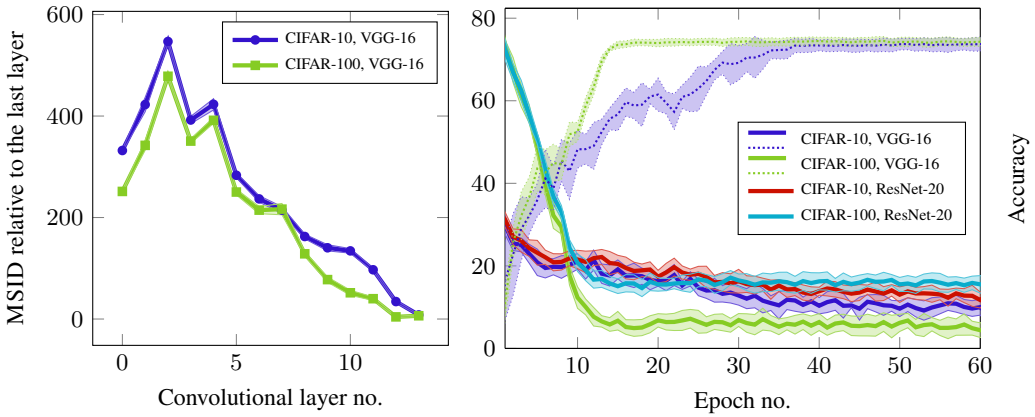

Figure 3: *(left)* MSID score across convolutional layers of the VGG-16 network on CIFAR-10 and CIFAR-100 datasets; *(right)* training progression in terms of accuracy (dotted) and MSID (solid) on CIFAR-10 and CIFAR-100 datasets for VGG-16 and ResNet-20, with respect to VGG-16.

the convolutional layers exhibit similar behavior; disentanglement appears as the network starts discriminating the manifold structure with a rapid MSID drop already after 6 convolutional blocks. The low variance across the 10 trained networks indicates stability in network structure.

We now examine the last network layers during training with different initializations. Figure 3 (right) plots the VGG16 validation errors and MSID scores relative to the final layer representations of *two* pretrained networks, VGG16 itelf with last layer dimension $d = 512$ and ResNet-20 with $d = 64$ and $\sim$50 times less parameters. We observe that even in such unaligned spaces, MSID correctly identifies the convergence point of the networks. Surprisingly, we find that, in terms of MSID, VGG-16 representations progress towards not only the VGG-16 final layer, but the ResNet-20 final layer representation as well; this result suggests that these networks of distinct architectures share similar final structures.

4.3 Interpretability of MSID

To understand how MSID operates, we investigate the behavior of heat kernel traces of different datasets that are normalized by a null model. Tsitsulin et al. [50] proposed a normalization by the heat kernel trace of an empty graph, which amounts to taking the average, rather than the sum, of the original heat kernel diagonal. However, this normalization is not an appropriate null model, as it ignores graph connectivity. We propose a heat kernel normalization by the *expected* heat kernel of an Erdős-Rényi graph. For that purpose, we need to approximate that graph's eigenvalues. Coja-Oghlan [16] proved that $\lambda_1 \leq 1 - c\bar{d}^{-1/2} \leq \lambda_2 \leq \lambda_n \leq 1 + c\bar{d}^{-1/2}$ for the core of the graph for some constant c. We have empirically found that $c = 2$ provides a tight approximation for random graphs. That coincides with the analysis of Chung et al. [13], who proved that $\lambda_n = (1 + o(1))2\bar{d}^{-1/2}$ if $d_{\min} \gg \sqrt{\bar{d}} \log^3 n$ even though in our case $d_{\min} = \bar{d} = k$. We thus estimate the spectrum of a random Erdős-Rényi graph as growing linearly between $\lambda_1 = 1 - 2\bar{d}^{-1/2}$ and $\lambda_n = 1 + 2\bar{d}^{-1/2}$, which corresponds to the underlying manifold being two-dimensional [50].

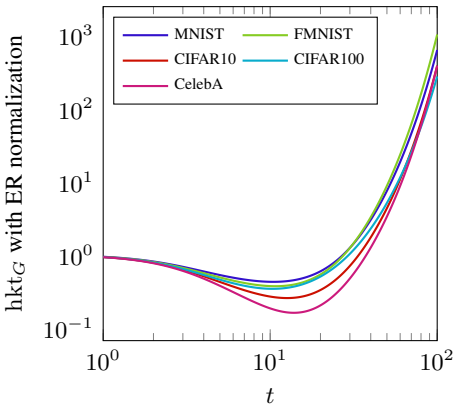

Figure 4: Plotting the heat trace reveals medium- and global-scale structure of datasets. Best viewed in color.

Figure 4 depicts the obtained normalized hkt_g for all datasets we work with. Again, we average results over 100 subsamples of $10k$ images each. For $t = 10$, i.e., at a medium scale, CelebA is most

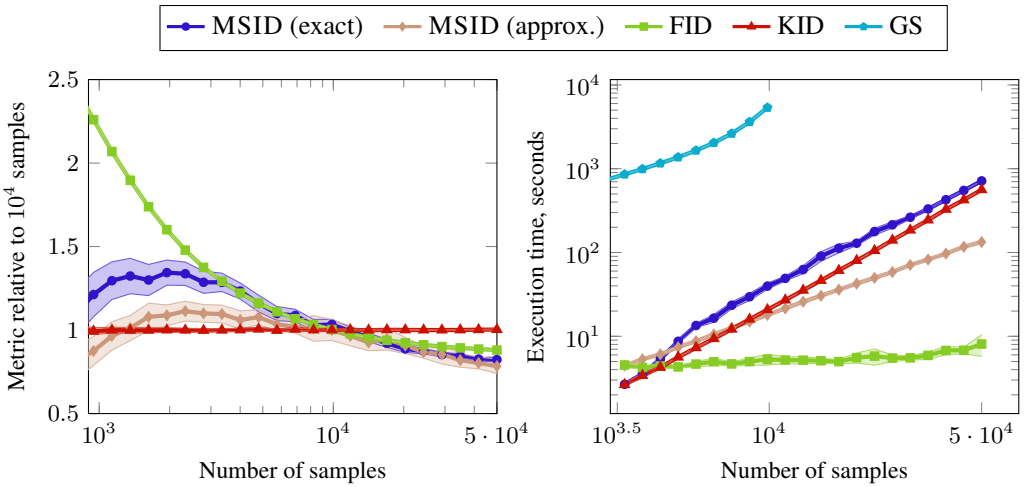

Figure 5: Stability and scalability experiment: *(left)* stability of FID, KID and MSID wrt. sample size on CIFAR-10 and CIFAR-100 dataset; *(right)* scalability of FID, KID and MSID wrt. sample size on synthetic datasets.

different from the random graph, while for large-scale t values, which capture global community structure, $\frac{\mathrm{dhkt}_g(t)}{\mathrm{d}t}$ reflects the approximate number of clusters in the data. Surprisingly, CIFAR-100 comes close to CIFAR-10 for large t values; we have found that this is due to the fact that the pre-trained Inception network does not separate the CIFAR-100 data classes well enough. We conclude that the heat kernel trace is interpretable if we normalize it with an appropriate null model.

4.4 Stability and scalability of MSID

In addition to the complexity analysis in Section 3.5, we assess the scaling and sample stability of MSID. Since MSID, like FID, is a lower bound to an optimal transport-based metric, we cannot hope for an unbiased estimator. However, we empirically verify, in Figure 5 (left), that MSID does not diverge too much with increased sample size. Most remarkably, we observe that MSID with approximate kNN [19] does not induce additional variance, while it diverges slightly further than the exact version as the number of samples grows.

In terms of scalability, Figure 5 (right) shows that the theoretical complexity is supported in practice. Using approximate kNN, we break the $\mathcal{O}(n^2)$ performance of KID. While FID's time complexity appears constant, in fact, its runtime is dominated by the $\mathcal{O}(d^3)$ matrix square root operation. Geometry score (GS) fails to perform scalably, as its runtime grows exponentially. Due to this prohibitive computational cost, we eschew further comparison with GS. Furthermore, as MSID distance is computed through a low-dimensional heat trace representation of the manifold, we can store HKT for future comparisons, thereby enhancing performance in the case of many-to-many comparisons.

5 Discussion and Future Work

We introduced MSID, the first intrinsic multi-scale method for comparing unaligned manifolds. Rooted in geometric theory, MSID provides valuable insights on the underlying data manifold. To achieve scalability, we develop a method for fast and accurate manifold comparison, grounded on the Stochastic Gauss Quadrature. We show the expressiveness of our method on the evaluation of generative models and the study of neural network representations. Since MSID allows comparing manifolds of diverse nature (e.g. different dimensionality, different representations), its applicability is not limited to the tasks we have evaluated. Multi-Scale Intrinsic Distance unveils important properties of neural networks and paves the way to the development of even more expressive techniques founded on similar geometric insights.

References

[1] Alessandro Achille and Stefano Soatto. Emergence of invariance and disentanglement in deep representations. *JMLR*, 2018.

[2] Ryan P Adams, Jeffrey Pennington, Matthew J Johnson, Jamie Smith, Yaniv Ovadia, Brian Patton, and James Saunderson. Estimating the spectral density of large implicit matrices. *arXiv preprint arXiv:1802.03451*, 2018.

[3] Alexander A. Alemi and Ian Fischer. GILBO: One metric to measure them all. In *NeurIPS*, 2018.

[4] A. Anonymous. Variance reduction in trace estimation of matrix functions. 2019.

[5] Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *ICML*, 2017.

[6] Martin Aumüller, Erik Bernhardsson, and Alexander Faithfull. ANN-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. *Information Systems*, 2019.

[7] Haim Avron and Sivan Toledo. Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix. *Journal of the ACM (JACM)*, 2011.

[8] Paul Balister, Béla Bollobás, Amites Sarkar, and Mark Walters. Connectivity of random k-nearest-neighbour graphs. *Advances in Applied Probability*, 2005.

[9] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, 2002.

[10] Mikhail Belkin and Partha Niyogi. Convergence of laplacian eigenmaps. In *NIPS*, pages 129–136, 2007.

[11] Mikołaj Bińkowski, Dougal J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD gans. In *ICLR*, 2018.

[12] Ali Borji. Pros and cons of gan evaluation measures. *Computer Vision and Image Understanding*, 179: 41–65, 2019.

[13] Fan Chung, Linyuan Lu, and Van Vu. The spectra of random graphs with given expected degrees. *Internet Mathematics*, 2004.

[14] Ed S. Coakley and Vladimir Rokhlin. A fast divide-and-conquer algorithm for computing the spectra of real symmetric tridiagonal matrices. *Applied and Computational Harmonic Analysis*, 34(3):379 – 414, 2013.

[15] Ronald R Coifman and Stéphane Lafon. Diffusion maps. *Applied and computational harmonic analysis*, 2006.

[16] Amin Coja-Oghlan. On the laplacian eigenvalues of g(n, p). *Combinatorics, Probability and Computing*, 2007.

[17] Emily L. Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. In *NIPS*, 2015.

[18] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *ICLR*, 2017.

[19] Wei Dong, Charikar Moses, and Kai Li. Efficient k-nearest neighbor graph construction for generic similarity measures. In *WWW*, 2011.

[20] Gene H Golub and Gérard Meurant. *Matrices, moments and quadrature with applications*. Princeton University Press, 2009.

[21] Gene H Golub and John H Welsch. Calculation of gauss quadrature rules. *Mathematics of computation*, 1969.

[22] Gene H Golub and John H Welsch. Calculation of gauss quadrature rules. *Mathematics of computation*, 23(106):221–230, 1969.

[23] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014.

[24] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander J. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13:723–773, 2012.

[25] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein GANs. In *NIPS*, 2017.

[26] Matthias Hein, Jean-Yves Audibert, and Ulrike von Luxburg. Graph laplacians and their convergence on random neighborhood graphs. *Journal of Machine Learning Research*, 2007.

[27] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *NIPS*, 2017.

[28] Marlis Hochbruck and Christian Lubich. On krylov subspace approximations to the matrix exponential operator. *SIAM Journal on Numerical Analysis*, 1997.

[29] MF Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

[30] Daniel Jiwoong Im, He Ma, Graham Taylor, and Kristin Branson. Quantitatively evaluating GANs with divergences proposed for training. In *ICLR*, 2018.

[31] Simon Jenni and Paolo Favaro. Noise-tempered generative adversarial networks, 2019. URL `https://openreview.net/forum?id=SygNooCqY7`.

[32] Valentin Khrulkov and Ivan V. Oseledets. Geometry score: A method for comparing generative adversarial networks. In *ICML*, 2018.

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

[34] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *ICML*, 2019.

[35] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. MMD GAN: Towards deeper understanding of moment matching network. In *NIPS*, 2017.

[36] David Lopez-Paz and Maxime Oquab. Revisiting classifier two-sample tests. In *ICLR*, 2017.

[37] Facundo Mémoli. A spectral notion of gromov–wasserstein distance and related methods. *Applied and Computational Harmonic Analysis*, 2011.

[38] Ari Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation. In *NeurIPS*, 2018.

[39] Youssef Mroueh, Chun-Liang Li, Tom Sercu, Anant Raj, and Yu Cheng. Sobolev GAN. In *ICLR*, 2018.

[40] Hariharan Narayanan and Sanjoy Mitter. Sample complexity of testing the manifold hypothesis. In *NIPS*, 2010.

[41] Augustus Odena, Jacob Buckman, Catherine Olsson, Tom B. Brown, Christopher Olah, Colin A. Raffel, and Ian J. Goodfellow. Is generator conditioning causally related to GAN performance? In *ICML*, 2018.

[42] Sung Woo Park and Junseok Kwon. Sphere generative adversarial network based on geometric moment matching. In *CVPR*, 2019.

[43] Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. SVCCA: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *NIPS*, 2017.

[44] Farbod Roosta-Khorasani and Uri Ascher. Improved bounds on sample size for implicit matrix trace estimators. *Foundations of Computational Mathematics*, 2015.

[45] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In *NIPS*, 2016.

[46] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015.

[47] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *Computer graphics forum*. Wiley Online Library, 2009.

[48] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[49] Daniel Ting, Ling Huang, and Michael Jordan. An analysis of the convergence of graph laplacians. In *ICML*, 2010.

[50] Anton Tsitsulin, Davide Mottin, Panagiotis Karras, Alexander M. Bronstein, and Emmanuel Müller. NetLSD: Hearing the shape of a graph. In *KDD*, 2018.

[51] Shashanka Ubaru, Jie Chen, and Yousef Saad. Fast estimation of tr(f(a)) via stochastic lanczos quadrature. *SIAM Journal on Matrix Analysis and Applications*, 2017.

[52] Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *ICML*, 2016.

[53] Liwei Wang, Lunjia Hu, Jiayuan Gu, Zhiqiang Hu, Yue Wu, Kun He, and John Hopcroft. Towards understanding learning representations: To what extent do different neural networks learn the same representation. In *Advances in Neural Information Processing Systems*, 2018.

[54] Herbert S Wilf. Mathematics for the physical sciences. 1962.

[55] Sharon Zhou, Mitchell Gordon, Ranjay Krishna, Austin Narcomey, Durim Morina, and Michael S Bernstein. HYPE: Human eYe perceptual evaluation of generative models. *arXiv preprint arXiv:1904.01121*, 2019.

Appendix

A Trace estimation error bounds

We will use the error of the Lanczos approximation of the action of the matrix exponential $f(\boldsymbol{\mathcal{L}})\mathbf{v} = \exp^{-t\boldsymbol{\mathcal{L}}}\mathbf{v}$ to estimate the error of the trace. We first rewrite quadratic form under summation in the trace approximation to a convenient form,

$$\mathbf{v}^\top f(\boldsymbol{\mathcal{L}})\mathbf{v} \approx \sum_{k=0}^{m} \tau_k^2 f(\lambda_k) = \sum_{k=0}^{m} [\mathbf{e}_1^\top \mathbf{u}_k]^2 f(\lambda_k) = \mathbf{e}_1^\top \mathbf{U} f(\Lambda)\mathbf{U}^\top \mathbf{e}_1 = \mathbf{e}_1^\top f(\mathbf{T})\mathbf{e}_1. \tag{7}$$

Because the Krylov subspace $\mathcal{K}_m(\boldsymbol{\mathcal{L}}, \mathbf{v})$ is built on top of vector $\mathbf{v}$ with $\mathbf{Q}$ as an orthogonal basis of $\mathcal{K}_m(\boldsymbol{\mathcal{L}}, \mathbf{v})$, i.e. $\mathbf{q}_0 = \mathbf{v}$ and $\mathbf{v} \perp \mathbf{q}_i$ for $i \in (1, \ldots, m-1)$, the following holds

$$\mathbf{v}^\top f(\boldsymbol{\mathcal{L}})\mathbf{v} \approx \mathbf{v}^\top \mathbf{Q} f(\mathbf{T})\mathbf{e}_1 = \mathbf{e}_1^\top f(\mathbf{T})\mathbf{e}_1. \tag{8}$$

Thus, the error in quadratic form estimate $\mathbf{v}^\top f(\boldsymbol{\mathcal{L}})\mathbf{v}$ is exactly the error of Lanczos approximation $f(\boldsymbol{\mathcal{L}})\mathbf{v} \approx \mathbf{Q} f(\mathbf{T})\mathbf{e}_1$. To obtain the error bounds, we use the Theorem 2 in [28], which we recite below.

Theorem 2 *Let $\boldsymbol{\mathcal{L}}$ be a real symmetric positive semi-definite matrix with eigenvalues in the interval $[0, 4\rho]$. Then the error in the m-step Lanczos approximation of $\exp^{-t\boldsymbol{\mathcal{L}}}\mathbf{v}$, i.e. $\epsilon_m = \| \exp^{-t\boldsymbol{\mathcal{L}}}\mathbf{v} - \mathbf{Q}_m \exp^{-t\mathbf{T}_m}\mathbf{e}_1 \|$, is bounded in the following ways:*

$$\epsilon_m \leq \begin{cases} 10e^{-m^2/(5\rho t)}, & \sqrt{4\rho t} \leq m \leq 2\rho t \quad (9a) \\ 10(\rho t)^{-1} e^{-\rho t} \left(\frac{e\rho t}{m}\right)^m, & m \geq 2\rho t \quad (9b) \end{cases}$$

Since $\mathbf{v}$ is a unit vector, thanks to Cauchy–Bunyakovsky–Schwarz inequality, we can upper-bound the error of the quadratic form approximation by the error of the $\exp^{-t\boldsymbol{\mathcal{L}}}\mathbf{v}$ approximation, i.e. $|\mathbf{v}^\top f(\boldsymbol{\mathcal{L}})\mathbf{v} - \mathbf{e}_1^\top \mathbf{U} f(\Lambda)\mathbf{U}^\top \mathbf{e}_1| \leq \| \exp^{-t\boldsymbol{\mathcal{L}}}\mathbf{v} - \mathbf{Q}_m \exp^{-t\mathbf{T}_m}\mathbf{e}_1 \| = \epsilon_m$.

Following the argumentation in [51], we obtain a condition on the number of Lanczos steps m by setting $\epsilon_m \leq \frac{\epsilon}{2} f_{min}(\lambda)$, where $f_{min}(\lambda)$ is the minimum value of f on $[\lambda_{min}, \lambda_{max}]$. We now derive the absolute error between the Hutchinson estimate of Equation (3) and the SLQ of Equation (6):

$$\begin{aligned} \Big| \operatorname{Tr}_{n_v}(f(\boldsymbol{\mathcal{L}})) - \Gamma \Big| &= \frac{n}{n_v} \left| \sum_{i=1}^{n_v} \mathbf{v}_i^\top f(\boldsymbol{\mathcal{L}})\mathbf{v}_i - \sum_{i=1}^{n_v} \mathbf{e}_1^\top f(\mathbf{T}^{(i)})\mathbf{e}_1 \right| \\ &\leq \frac{n}{n_v} \sum_{i=1}^{n_v} \left| \mathbf{v}_i^\top f(\boldsymbol{\mathcal{L}})\mathbf{v}_i - \mathbf{e}_1^\top f(\mathbf{T}^{(i)})\mathbf{e}_1 \right| \\ &\leq \frac{n}{n_v} \sum_{i=1}^{n_v} \epsilon_m = n\epsilon_m, \end{aligned}$$

where $\mathbf{T}^{(i)}$ is the tridiagonal matrix obtained with Lanczos algorithm with starting vector $\mathbf{v}_i$.

$$\Big| \operatorname{Tr}_{n_v} f(\boldsymbol{\mathcal{L}}) - \Gamma \Big| \leq n\epsilon_m \leq \frac{n\epsilon}{2} f_{min}(\lambda) \leq \frac{\epsilon}{2} \operatorname{Tr}(f(\boldsymbol{\mathcal{L}})), \tag{10}$$

Finally, we formulate SLQ as an (ϵ, δ) estimator,

$$\begin{aligned} 1 - \delta &\leq \Pr\left[\Big| \operatorname{Tr}(f(\boldsymbol{\mathcal{L}})) - \operatorname{Tr}_{n_v}(f(\boldsymbol{\mathcal{L}})) \Big| \leq \frac{\epsilon}{2} \Big| \operatorname{Tr}(f(\boldsymbol{\mathcal{L}})) \Big| \right] \\ &\leq \Pr\left[\Big| \operatorname{Tr}(f(\boldsymbol{\mathcal{L}})) - \operatorname{Tr}_{n_v}(f(\boldsymbol{\mathcal{L}})) \Big| + \Big| \operatorname{Tr}_{n_v}(f(\boldsymbol{\mathcal{L}})) - \Gamma \Big| \leq \frac{\epsilon}{2} \Big| \operatorname{Tr}(f(\boldsymbol{\mathcal{L}})) \Big| + \frac{\epsilon}{2} \Big| \operatorname{Tr}(f(\boldsymbol{\mathcal{L}})) \Big| \right] \\ &\leq \Pr\left[\Big| \operatorname{Tr}(f(\boldsymbol{\mathcal{L}})) - \Gamma \Big| \leq \epsilon \Big| \operatorname{Tr}(f(\boldsymbol{\mathcal{L}})) \Big| \right], \end{aligned}$$

For the normalized Laplacian $\boldsymbol{\mathcal{L}}$, the minimum eigenvalue is 0 and $f_{\min}(0) = \exp(0) = 1$, hence $\epsilon_m \leq \frac{\epsilon}{2}$, and the eigenvalue interval has $\rho = 0.5$. We can thus derive the appropriate number of Lanczos steps m to achieve error ϵ,

$$\epsilon \leq \begin{cases} 20e^{-m^2/(2.5t)}, & \sqrt{2t} \leq m \leq t \quad (11a) \\ 40t^{-1}e^{-0.5t}\left(\frac{0.5et}{m}\right)^m, & m \geq t \quad (11b) \end{cases}$$

Figure 6 shows the tightness of the bound for the approximation of the matrix exponential action on vector $\mathbf{v}$, $\epsilon_m = \| \exp(-t\boldsymbol{\mathcal{L}}) - \mathbf{Q}_m \exp(-t\mathbf{T}_m)\mathbf{e}_1 \|$. We can see that for most of the temperatures t, very few Lanczos steps m are sufficient, i.e. we can set $m = 10$. However, the error from the Hutchinson estimator dominates the overall error. Figure 7 shows the error of trace estimation does not change with m and for $t = 0.1$ is around 10^{-3}. In case of a Rademacher $p(\mathbf{v})$, the bound on the number of random samples is $n_v \geq \frac{6}{\epsilon^2}\log(2/\delta)$ [44]. Employing 10k vectors results in the error bound of roughly 10^{-2}. In practice, we observe the performance much better than given by the bound, see Figure 7.

One particular benefit of small m value is that we do not have to worry about the orthogonality loss in the Lanczos algorithm which often undermines its convergence. Since we do only a few Lanczos iterations, the rounding errors hardly accumulate causing little burden in terms of orthogonality loss between the basis vectors of the Krylov subspace.

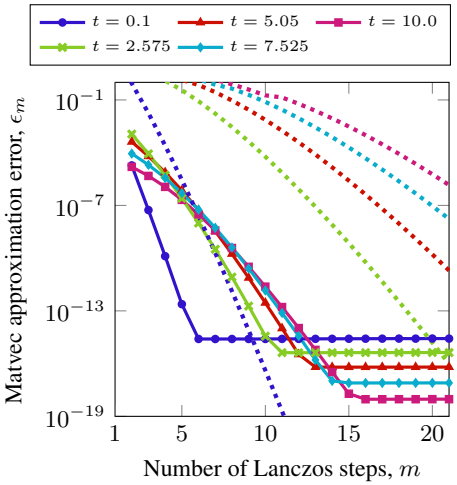

Figure 6: Errors (solid) and error bounds (dotted) for the approximation of matrix exponential action with varying temperature t.

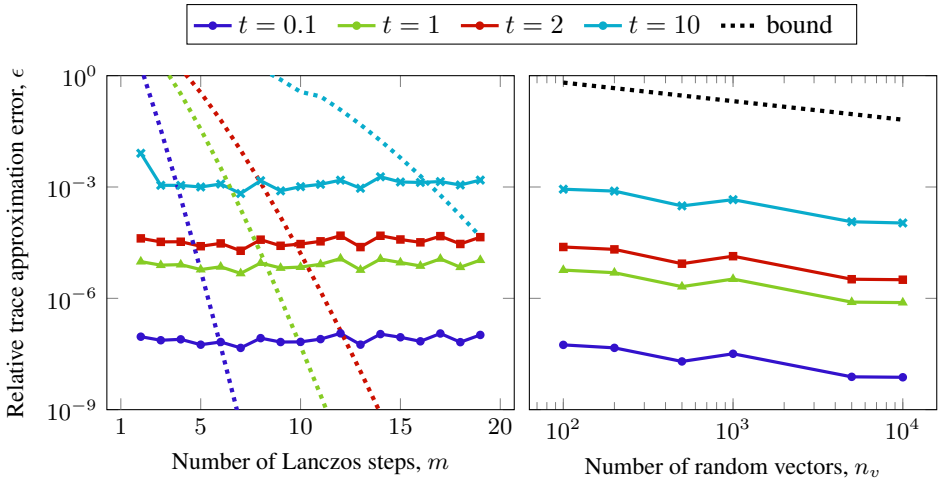

Figure 7: Trace estimation errors (solid) and error bounds (dotted) for: *(left)* the number of Lanczos steps m with fixed number of random vectors $n_v = 100$; *(right)* the number of random vectors n_v in Hutchinson estimator with fixed number of Lanczos steps $m = 10$. Lines correspond to varying temperatures t.

B Variance reduction

We apply the variance reduction technique from [4]. The idea is to use Taylor expansion to substitute a part of the trace estimate with its easily computed precise value,

$$\mathrm{Tr}(\exp(-t\boldsymbol{\mathcal{L}})) = \mathtt{slq}\Big[\exp(-t\boldsymbol{\mathcal{L}}) - (\mathbf{I} - t\boldsymbol{\mathcal{L}} + \frac{t^2\boldsymbol{\mathcal{L}}^2}{2})\Big] + \mathrm{Tr}(\mathbf{I} - t\boldsymbol{\mathcal{L}} + \frac{t^2\boldsymbol{\mathcal{L}}^2}{2}) \tag{12}$$

$$= \mathtt{slq}\Big[\exp(-t\boldsymbol{\mathcal{L}}) - (\mathbf{I} - t\boldsymbol{\mathcal{L}} + \frac{t^2\boldsymbol{\mathcal{L}}^2}{2})\Big] + n + \mathrm{Tr}(-t\boldsymbol{\mathcal{L}}) + \frac{t^2\|\boldsymbol{\mathcal{L}}\|_F^2}{2} \tag{13}$$

$$= \mathtt{slq}\Big[\exp(-t\boldsymbol{\mathcal{L}})\Big] + \mathtt{slq}\Big[t\boldsymbol{\mathcal{L}}\Big] - \mathtt{slq}\Big[\frac{t^2\boldsymbol{\mathcal{L}}^2}{2})\Big] - tn + \frac{t^2\|\boldsymbol{\mathcal{L}}\|_F^2}{2}, \tag{14}$$

where we use the fact that $\|\boldsymbol{\mathcal{L}}\|_F = \sqrt{\mathrm{Tr}(\boldsymbol{\mathcal{L}}^\top\boldsymbol{\mathcal{L}})}$ and that the trace of normalized Laplacian is equal to n. It does reduce the variance of the trace estimate for smaller temperatures $t \leq 1$.

To obtain this advantage over the whole range of t, we utilize the following variance reduction form:

$$\mathrm{Tr}(\exp(-t\boldsymbol{\mathcal{L}})) = \mathtt{slq}\Big[\exp(-t\boldsymbol{\mathcal{L}}) - (\mathbf{I} - \alpha t\boldsymbol{\mathcal{L}})\Big] + n(1 - \alpha t), \tag{15}$$

where there exists an alpha that is optimal for every t, namely setting $\alpha = 1/\exp(t)$. We can see the variance reduction that comes from this procedure in the Figure 8.

C Experimental settings

We train all our models on a single server with NVIDIA V100 GPU with 16Gb memory and 2×20 core Intel E5-2698 v4 CPU. For the experiment summarized in Table 1 in the Section 4.1 we train WGAN and WGAN-GP models on 4 datasets: MNIST, FashionMNIST, CIFAR10 and CelebA and sample 10k samples, $\mathbf{Y}$, from each of the GANs. We uniformly subsample 10k images from the original datasets, $\mathbf{X}$, and compute the MSID, KID and FID scores between $\mathbf{X}$ and $\mathbf{Y}$. We report the mean as well as the 99% confidence interval across 100 runs.

Below we report the architectures, hyperparameters and generated samples of the models used for the experiments. We train each of the GANs for 200 epochs on MNIST, FMNIST and CIFAR-10, and for 50 epochs on CelebA dataset. For WGAN we use RMSprop optimizer with learning rate of 5×10^{-5}. For WGAN-GP we use Adam optimizer with learning rate of 10^{-4}, $\beta_1 = 0.9, \beta_2 = 0.999$.

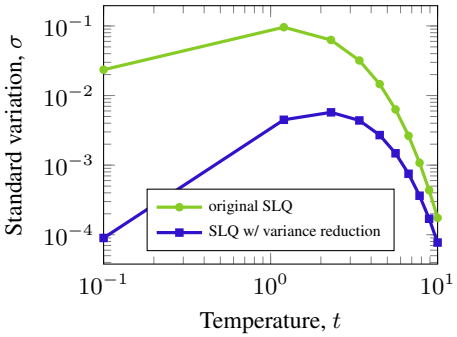

Figure 8: Variance of the trace estimate.

D Graph example

Figure 9 provides visual proof that the 5NN graph reflects the underlying manifold structure of the CIFAR-10 dataset. Clusters in the graph exactly correspond to CIFAR-10 classes.

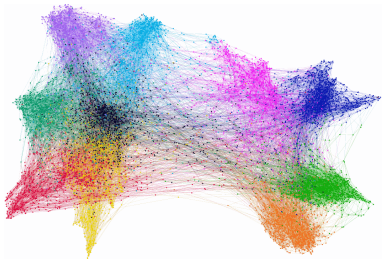

Figure 9: CIFAR-10 graph colored with true class labels.

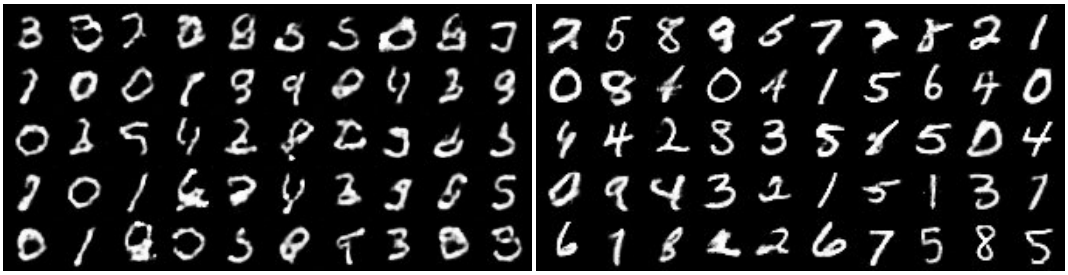

Figure 10: MNIST samples (left: WGAN, right: WGAN-GP)

Figure 11: FashionMNIST samples (left: WGAN, right: WGAN-GP)

Figure 12: CIFAR-10 samples (left: WGAN, right: WGAN-GP)

Figure 13: CelebA samples (left: WGAN, right: WGAN-GP)

MNIST WGAN

```
ConvGenerator(
  (latent_to_features): Sequential(
    (0): Linear(in_features=100, out_features=512, bias=True)
    (1): ReLU()
  )
  (features_to_image): Sequential(
    (0): ConvTranspose2d(128, 64, kernel_size=(4, 4),
                         stride=(2, 2), padding=(1, 1))
    (1): ReLU()
    (2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True)
    (3): ConvTranspose2d(64, 32, kernel_size=(4, 4),
                         stride=(2, 2), padding=(1, 1))
    (4): ReLU()
    (5): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True)
    (6): ConvTranspose2d(32, 16, kernel_size=(4, 4),
                         stride=(2, 2), padding=(1, 1))
    (7): ReLU()
    (8): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True)
    (9): ConvTranspose2d(16, 1, kernel_size=(4, 4),
                         stride=(2, 2), padding=(1, 1))
    (10): Sigmoid()
  )
)

ConvDiscriminator(
  (image_to_features): Sequential(
    (0): Conv2d(1, 16, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (1): LeakyReLU(negative_slope=0.2)
    (2): Conv2d(16, 32, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (3): LeakyReLU(negative_slope=0.2)
    (4): Conv2d(32, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (5): LeakyReLU(negative_slope=0.2)
    (6): Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (7): Sigmoid()
  )
  (features_to_prob): Sequential(
    (0): Linear(in_features=512, out_features=1, bias=True)
    (1): Sigmoid()
  )
)
```

MNIST WGAN-GP, FMNIST (WGAN, WGAN-GP)

```
MNISTGenerator(
  (block1): Sequential(
    (0): ConvTranspose2d(256, 128, kernel_size=(5, 5), stride=(1, 1))
    (1): ReLU(inplace)
  )
  (block2): Sequential(
    (0): ConvTranspose2d(128, 64, kernel_size=(5, 5), stride=(1, 1))
    (1): ReLU(inplace)
  )
  (deconv_out): ConvTranspose2d(64, 1, kernel_size=(8, 8), stride=(2, 2))
  (preprocess): Sequential(
    (0): Linear(in_features=128, out_features=4096, bias=True)
    (1): ReLU(inplace)
  )
  (sigmoid): Sigmoid()
)

MNISTDiscriminator(
  (main): Sequential(
    (0): Conv2d(1, 64, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))
    (1): ReLU(inplace)
    (2): Conv2d(64, 128, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))
    (3): ReLU(inplace)
    (4): Conv2d(128, 256, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))
    (5): ReLU(inplace)
  )
  (output): Linear(in_features=4096, out_features=1, bias=True)
)
```

CIFAR-10 (WGAN, WGAN-GP)

```
CIFARGenerator(
  (preprocess): Sequential(
    (0): Linear(in_features=128, out_features=4096, bias=True)
    (1): BatchNorm1d(4096, eps=1e-05, momentum=0.1, affine=True)
    (2): ReLU(inplace)
  )
  (block1): Sequential(
    (0): ConvTranspose2d(256, 128, kernel_size=(2, 2), stride=(2, 2))
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True)
    (2): ReLU(inplace)
  )
  (block2): Sequential(
    (0): ConvTranspose2d(128, 64, kernel_size=(2, 2), stride=(2, 2))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True)
    (2): ReLU(inplace)
  )
  (deconv_out): ConvTranspose2d(64, 3, kernel_size=(2, 2), stride=(2, 2))
  (tanh): Tanh()
)

CIFARDiscriminator(
  (main): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (1): LeakyReLU(negative_slope=0.01)
    (2): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (3): LeakyReLU(negative_slope=0.01)
    (4): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (5): LeakyReLU(negative_slope=0.01)
  )
  (linear): Linear(in_features=4096, out_features=1, bias=True)
)
```

CelebA (WGAN, WGAN-GP)

```
CelebaGenerator(
  (preprocess): Sequential(
    (0): Linear(in_features=128, out_features=8192, bias=True)
    (1): BatchNorm1d(8192, eps=1e-05, momentum=0.1, affine=True)
    (2): ReLU(inplace)
  )
  (block1): Sequential(
    (0): ConvTranspose2d(512, 256, kernel_size=(5, 5), stride=(2, 2),
                         padding=(2, 2), output_padding=(1, 1), bias=False)
    (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True)
    (2): ReLU(inplace)
  )
  (block2): Sequential(
    (0): ConvTranspose2d(256, 128, kernel_size=(5, 5), stride=(2, 2),
                         padding=(2, 2), output_padding=(1, 1), bias=False)
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True)
    (2): ReLU(inplace)
  )
  (block3): Sequential(
    (0): ConvTranspose2d(128, 64, kernel_size=(5, 5), stride=(2, 2),
                         padding=(2, 2), output_padding=(1, 1), bias=False)
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True)
    (2): ReLU(inplace)
  )
  (deconv_out): ConvTranspose2d(64, 3, kernel_size=(5, 5), stride=(2, 2),
                                padding=(2, 2), output_padding=(1, 1))
  (tanh): Tanh()
)

CelebaDiscriminator(
  (main): Sequential(
    (0): Conv2d(3, 64, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))
    (1): LeakyReLU(negative_slope=0.01)
    (2): Conv2d(64, 128, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))
    (3): LeakyReLU(negative_slope=0.01)
    (4): Conv2d(128, 256, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))
    (5): LeakyReLU(negative_slope=0.01)
    (6): Conv2d(256, 512, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))
    (7): LeakyReLU(negative_slope=0.01)
    (8): Conv2d(512, 1, kernel_size=(4, 4), stride=(1, 1))
  )
)
```