Latent Space Geometry Search for Neural Networks via Random Walks on Product Manifolds

Adam Mehdi
Columbia University
adm2243@columbia.edu

Anupam Bhakta
Columbia University
ab5494@columbia.edu

Kevin Z. Qiu
Columbia University
kzq2000@columbia.edu

Abstract

This paper investigates the impact of latent space geometry on the performance of neural networks. We aim to identify the optimal latent geometry for a given task by searching over a graph space defined on product manifolds, where each node represents a unique latent space geometry and edges are weighted by the inverse of a modified Gromov-Hausdorff distance. Our approach follows Sáez de Ocáriz Borde et al. [2024], but we diverge in the choice of graph search method, opting for random walks instead of Bayesian optimization. We leverage random walks and their connections to spectral clustering to explore and understand the graph search space. Experiments on autoencoders trained on the MNIST dataset [Deng, 2012] evaluate the effectiveness of different random walk techniques, including weighted versus unweighted walks and nonbacktracking versus backtracking walks. The results demonstrate that backtracking walks make a noticeable difference in performance compared to a naive random walk, particularly over a smaller number of iterations. There is, however, less of a difference between the performance of both methods using weighted and unweighted edges in the search space. The code for the method and experiments is provided on our GitHub.

1 Introduction

Neural networks have emerged as a fundamental tool in machine learning. At the core of their functionality lies the concept of latent space, an abstract, lower-dimensional space where data is mapped by the network. This latent space captures essential features and relationships within the data, facilitating tasks such as classification, regression, and generation [Goodfellow et al., 2016]. Neural networks transform high-dimensional input data into a compressed and abstract representation in the latent space, discarding noise and irrelevant details while preserving meaningful patterns and structures [Hinton and Salakhutdinov, 2006].

A central dogma of geometric data analysis is that data inherently possesses geometry [Rabadán and Blumberg, 2019]. For instance, hierarchical structures found in phylogenetic trees can be effectively represented using hyperbolic geometry, which preserves generational information without clutter [Billera et al., 2001]. Similarly, cyclic data in time series can be captured by spherical geometry, which maintains rotational symmetries [Bronstein et al., 2006]. Selecting an appropriate latent space geometry based on these intrinsic properties of data can significantly enhance the performance of machine learning models across various domains [Shukla et al., 2018]. By aligning the latent space geometry with the data's intrinsic geometry, more efficient representations can be achieved, leading to improved model generalization without relying on increased network capacity. This approach alleviates issues related to over-parameterization and feature redundancy [Fei et al., 2023].

Despite the importance of aligning latent space geometry with data geometry, identifying the inherent geometry of data remains a challenging task. Current methods often resort to trying random geometries, a process typically performed heuristically and involving a random search [Hauberg et al., 2012]. This approach lacks a principled way of searching for an optimal geometry that best captures the intrinsic properties of the data. In this paper, we adapt the work of Sáez de Ocáriz Borde et al. [2024], who introduced a framework for neural latent geometry search. While they favor Bayesian optimization as the search strategy, we opt for random walks due to their geometric interpretations and simpler exploration of the search space.

Our contribution lies in the implementation of a principled approach for searching an optimal latent space geometry. We construct a graph search space over product manifolds, where the weights between manifolds are determined by the inverse Gromov-Hausdorff distance. To efficiently compute the distances between arbitrary product manifolds, we utilize precomputed values for 2-dimensional model spaces and employ the Hungarian algorithm for optimal matching. We define several search methods over the graph search space, including an unweighted random walk over the pruned graph, a random walk with transition probabilities weighted by the inverse modified Gromov-Hausdorff distance, and a random walk with backtracking as an approximation to discretized stochastic gradient descent. We evaluate these search methods using an image autoencoder on the MNIST dataset.

2 Background

2.1 Product Manifolds

A product manifold is a mathematical structure formed by combining two or more simpler manifolds through the Cartesian product operation. Given two manifolds M and N, their product manifold, denoted as M × N, is the set of all ordered pairs (m, n), where m is a point from manifold M and n is a point from manifold N. The dimensionality of the product manifold is the sum of the dimensions of its component manifolds. Product manifolds inherit important properties from their component manifolds, such as the exponential map and geodesic distance. The exponential map of a product manifold is a diffeomorphism between its tangent space and the manifold itself, allowing for a locally linear structure. The geodesic distance on a product manifold can be computed using the geodesic distances of its component manifolds [Lee and Lee, 2012], as described by the equation:

$$d_{(M\times N)}((m_1,n_1),(m_2,n_2)) = \sqrt{d_M(m_1,m_2)^2 + d_N(n_1,n_2)^2}. \tag{1}$$

The use of product manifolds allows for the construction of more complex topological and geometric structures by combining simpler, well-understood manifolds, such as those with hyperbolic, Euclidean, and spherical geometries. These component manifolds serve as building blocks, each with its unique properties and advantages. Hyperbolic manifolds, with their constant negative curvature, are particularly useful for modeling hierarchical structures or networks exhibiting exponential growth properties. Euclidean manifolds are applicable to spaces where straight-line distance is directly relevant. Spherical manifolds, with their constant positive curvature, are suitable for representing closed, bounded systems or cyclical data. However, we note that not all manifold types or topological spaces can be neatly decomposed into or represented as product manifolds, such as those with twisting or nonuniform curvature [Bronstein et al., 2017]. The product manifold is used as an efficient representation for searching through a large class of manifolds [Sáez de Ocáriz Borde et al., 2024].

2.2 Gromov-Haustorff Distance

The Gromov-Hausdorff distance is a metric that allows for the comparison of metric spaces, even when they are not embedded in the same ambient space. It extends the concept of the Hausdorff distance, which is limited to comparing subsets within the same metric space. Given two metric spaces (X, d_X) and (Y, d_Y), the Hausdorff distance between subsets $A, B \subseteq X$ is defined as:

$$d_{H(A,B)} = \max(\min_{\epsilon}(B \subseteq A_\epsilon), \min_{\epsilon}(A \subseteq B_\epsilon)) \tag{2}$$

where A_ϵ and B_ϵ denote the ϵ-fattening of sets A and B, respectively. The Hausdorff distance is the minimum ϵ such that the ϵ-fattening of each set covers the other set. While the Hausdorff distance is

Table 1: Exponential map and distance functions for different geometries. x_p represents the base point on the manifold, v is the tangent vector, and K_S and K_H are the curvatures of the spherical and hyperbolic manifolds, respectively.

Geometry	Exponential Map $\exp_{x_p}(v)$	Distance Function $d(x, y)$
Euclidean	v	$\|x - y\|$
Spherical	$\cos(\sqrt{K_S}\|v\|)x_p + \sin(\sqrt{K_S}\|v\|)\frac{v}{\sqrt{K_S}\|v\|}$	$\frac{\arccos(K_S\langle x,y\rangle)}{\sqrt{K_S}}$
Hyperbolic	$\cosh(\sqrt{-K_H}\|v\|)x_p + \sinh(\sqrt{-K_H}\|v\|)\frac{v}{\sqrt{-K_H}\|v\|}$	$\frac{\text{arccosh}(K_H\langle x,y\rangle)}{\sqrt{-K_H}}$

a metric, it is restricted to comparing subsets within the same metric space and is sensitive to rigid transformations, such as isometries.

To overcome these limitations, the Gromov-Hausdorff distance is used:

$$d_{GH}((X, d_X), (Y, d_Y)) = \inf_{\theta_1 : X \to Z, \theta_2 : Y \to Z} d_H(\theta_1(X), \theta_2(Y)) \tag{3}$$

where θ_1 and θ_2 are isometries mapping X and Y into a common metric space Z. The Gromov-Hausdorff distance finds the infimum over all possible isometric embeddings of X and Y into Z, and then computes the Hausdorff distance between the embedded spaces.

In practice, computing the Gromov-Hausdorff distance can be challenging, so approximations are used. In addition, when comparing manifolds that are not defined over the same ambient space such as product manifolds with different signatures, the manifolds must be mapped to a common ambient space $\mathbb{E}^{6n-6}$, where n is the maximum dimension of the two product spaces, before applying the Gromov-Hausdorff distance [de Ocariz Borde et al., 2023].

3 Method

3.1 Enforcing Latent Geometry

To enforce a specific geometry on the latent space, we utilize the exponential map, a function that maps a tangent vector from the tangent space at a point on a manifold to a point on the manifold itself. The exponential map preserves the local structure of the manifold, allowing us to map a Euclidean latent vector to a point on the desired manifold. By projecting the latent vector onto the manifold using the exponential map, we effectively force the latent representation to lie on the manifold, causing the latent space to inherit the geometric properties of the manifold, such as its curvature and distance metric [Nickel and Kiela, 2017].

The exponential map equations, as presented in Table 1, provide a differentiable means of projecting latent vectors onto the desired manifold, thus enforcing the geometric properties of the manifold on the latent space. This differentiability is necessary for gradient-based optimization such as backpropagation Plaut and Hinton [1987]. The distance functions, also summarized in Table 1, allow for the computation of geodesic distances between points on the manifold, which is useful for various tasks such as similarity measurement and clustering.

The intuition behind why projecting the latent vector is sufficient to assume the desired geometry stems from the fact that the projection operation constrains the latent representation to the manifold. This constraint ensures that the latent space adheres to the geometric properties of the manifold, such as the curvature and distance metric, which in turn influences the model's learning and generalization capabilities [Nickel and Kiela, 2017].

While our current work focuses on exploring latent geometries for image autoencoders, the concept of enforcing specific geometries on the latent space can be extended to various other tasks and domains. In natural language processing, word or sentence embeddings can be constrained to lie on a manifold by applying the exponential map to the latent vectors, similar to our approach in image autoencoders. This projection onto a manifold with appropriate geometric properties could potentially capture the underlying semantic structure more effectively [Jawanpuria et al., 2019]. In graph representation learning, hyperbolic or spherical geometry can be enforced on the latent space by using the corresponding exponential maps during the learning process [Hamilton et al., 2017]. For

Table 2: Gromov-Haustorff distances between different model spaces.

Model Space 1	Model Space 2	Gromov-Hausdorff Distance
$\mathbb{E}^2$	$\mathbb{S}^2$	0.23
$\mathbb{E}^2$	$\mathbb{H}^2$	0.77
$\mathbb{S}^2$	$\mathbb{H}^2$	0.84

generative models, such as variational autoencoders or generative adversarial networks, the choice of latent geometry can be enforced by modifying the prior distribution and the corresponding sampling techniques to match the desired manifold [Chadebec and Allassonnière, 2022]. Investigating the impact of enforcing latent geometries on different tasks is an future research which may lead to improved representations across a wide range of applications.

3.2 Defining Graph Search Space

To define the search space for the optimal latent geometry, we construct a graph where each node represents a product manifold, and the edges between nodes are weighted by the inverse Gromov-Haustorff distance between the corresponding product manifolds. The distances between the base 2-dimensional model spaces ($\mathbb{E}^2, \mathbb{H}^2, \mathbb{S}^2$ are obtained from de Ocariz Borde et al. [2023], which provides a modified computation of the Gromov-Hausdorff distance. The method involves sampling distributed points from the model spaces, mapping them to $\mathbb{E}^{6n-6}$, which has been shown to be able to hold manifolds in $\mathbb{E}^n, \mathbb{S}^n$, and $\mathbb{H}^n$ without distortion, and applying the Gromov-Hausdorff distance in this higher-dimensional space de Ocariz Borde et al. [2023]. Due to the intricate mathematical specifications required to map manifolds hyperbolic space to a higher-dimensional Euclidean space, we directly use the distance values for the 2-dimensional model spaces. We extend the methodology to higher-dimensional product manifolds by combining multiple 2-dimensional model spaces as components. For example, a four-dimensional spherical space $\mathbb{S}^4$ is represented as the product manifold $\mathbb{S}^2 \times \mathbb{S}^2$.

In addition to restricting model space dimensionality to 2, edges are only defined between product manifolds that differ in a single model space component, ensuring a consistent and structured search space. For instance, $\mathbb{E}^2 \times \mathbb{H}^2$ and $\mathbb{E}^2 \times \mathbb{S}^2$ would be connected with an edge weighted by the inverse of the Gromov-Hausdorff distance between $\mathbb{H}^2$ and $\mathbb{S}^2$ respectively, while $\mathbb{S}^2 \times \mathbb{S}^2$ and $\mathbb{E}^2 \times \mathbb{E}^2$ would have no direct connection in the graph [Sáez de Ocáriz Borde et al., 2024]. In this formulation, the graph search space is defined over product manifolds with the same number of component manifolds n_p. These assumptions allow us to compute the distance between arbitrary product manifolds rapidly using the table of Gromov-Haustorff distances in 2 and the Hungarian algorithm as follows.

The computation begins by checking if the two product manifolds have the same number of component manifolds. If not, a distance of 0 is assigned between them, indicating no direct connection. Next, the types of component manifolds in each product manifold are identified, such as Euclidean (E2), spherical (S2), or hyperbolic (H2) in a 2-dimensional case. A cost matrix is then created based on the Gromov-Hausdorff distances between the component manifold types, which can be obtained from a pre-computed table like Table 2. The Hungarian algorithm is applied to the cost matrix to find the minimum weight perfect matching between the component manifolds of the two product manifolds, efficiently finding the optimal alignment that minimizes the total distance between the matched pairs. The weight between the two product manifolds is calculated as the sum of the distances between the optimally matched component manifolds, and the inverse of this weight is returned as the final distance between the product manifolds. Taking the inverse ensures that a higher weight corresponds to a smaller distance and vice versa.

We can define the total size of the graph search space as an expression of the number of distinct model spaces (n_s) and the total number of model spaces used to form the product manifold (n_p):

$$s = \sum_{i=1}^{n_p} \binom{n_s + i - 1}{i}. \tag{4}$$

This formula is derived from the concept of combinations with repetition, where the order of selection does not matter, and each model space can be chosen multiple times. The binomial coefficient

$\binom{n_s+i-1}{i}$ counts the number of ways to choose i model spaces from a set of n_s distinct model spaces, allowing for repetition.

The size of the graph search space grows exponentially as the number of model spaces (n_p) increases. Consequently, practical limitations often restrict the search to smaller latent spaces to maintain feasibility. In this work, we restrict $n_p = 5$, which limits us to latent dimension 10. Future work may focus on alleviating the combinatorial explosion of the graph search space or principled heuristics of identifying fruitful regions of the graph search space.

3.3 Searching over Graph Search Space

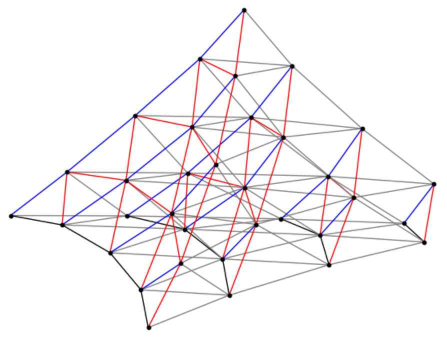
To explore the graph search space and identify the optimal latent geometry, we employ random walks, a simple yet effective method to traverse the graph with geometric interpretations [Göbel and Jagers, 1974]. In this approach, we define the performance of a product manifold as a metric evaluated on a validation set for a model trained with the given latent geometry. The performance of a random walk is the minimum of such evaluations on the geometries that make it up. We note that [Sáez de Ocáriz Borde et al., 2024] use Bayesian Optimization to traverse the graph search space. However, this approach, which works by building a probabilistic surrogate model and strategically sampling points based on acquisition functions, to be so complex as to detract from a geometric understanding of the graph search space [Frazier, 2018]. Random walks on graphs have been shown to have a close connection to spectral clustering [Von Luxburg, 2007]. In spectral clustering, the goal is to find a partition of the graph $G = (V, E)$ such that the random walk stays within the same cluster for a long time and rarely jumps between clusters. This intuition aligns well with the idea of finding a balanced partition with a low cut, as such a partition would limit the opportunities for the random walk to transition between clusters [Ding et al., 2001].

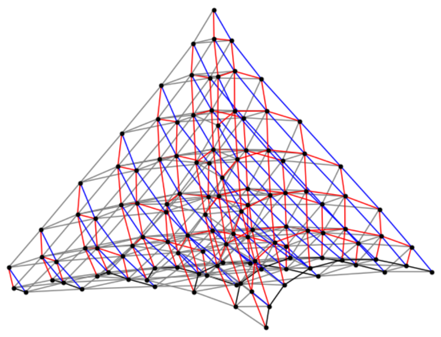
To perform random walks on our graph search space, we first normalize the inverse Gromov-Hausdorff weighted adjacency matrix A into a stochastic transition matrix P. The transition probability p_{ij} of jumping from a product manifold i to a product manifold j in one step is proportional to the inverse of their Gromov-Hausdorff distance, i.e., $p_{ij} = w_{ij}/d_i$, where w_{ij} is the inverse Gromov-Hausdorff distance and d_i is the degree of node i, defined as $d_i = \sum_j w_{ij}$. This normalization ensures that the random walk is more likely to step to closer product manifolds in the graph search space, enforcing that long distances do not matter.

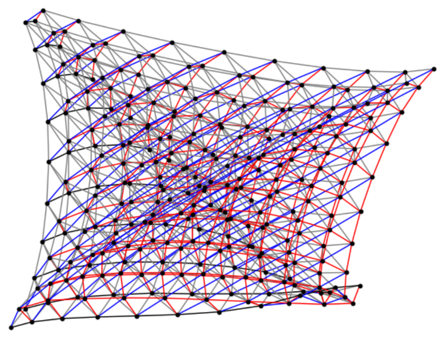
The relationship between random walks and spectral clustering provides a geometric interpretation of the random walk process on our graph search space. The transition matrix P is closely related to the graph Laplacian $L = D - A$, where D is the diagonal degree matrix with $D_{ii} = d_i$. The spectrum of L, consisting of its eigenvalues and corresponding eigenvectors, encodes the geometry of the graph G. In particular, the multiplicity of the eigenvalue $\lambda = 0$ equals the number of connected components in the graph [Geometric Data Analysis Class Notes]. By normalizing the adjacency matrix based on the inverse Gromov-Hausdorff distances, we essentially encourage the random walk to stay within clusters of product manifolds that are geometrically similar. In our experiments, we sample from different product manifolds as a starting point to encourage the random walk process to explore different regions of the graph search space. We expect performance within clusters to differ less relative to performance across clusters.

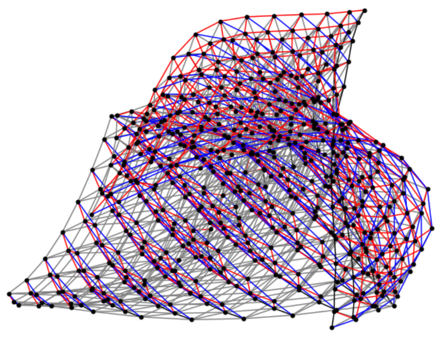
Inspired by stochastic gradient descent, we also experiment with random walk with backtracking [Amari, 1993]. If a random walk step leads to a decrease in performance, we backtrack to the previous product manifold and step in a new direction. This process can be seen as a discrete approximation to stochastic gradient descent, where instead of always moving in the direction of steepest descent, we move in any direction that decreases objective function [Stephan et al., 2017]. However, this approach can get stuck in local minima if a geometry is the best-performing among its immediate neighbors in the graph. The effectiveness of random walk with backtracking relies on the assumption that the graph search space is smooth and the performance metric decreases gradually [Andradóttir, 1995]. We hypothesize that the performance of the random walk with backtracking method will provide insights into the geometric properties of the graph search space with respect to model performance on a given dataset.

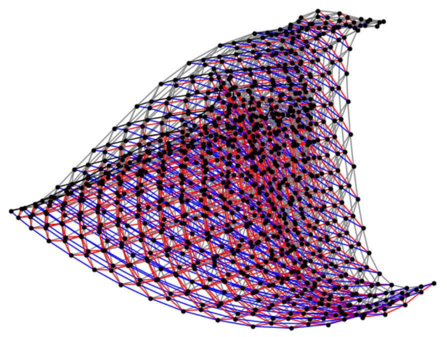
Figure 1: Sample graph search spaces for $n_s = 3$ and different values of n_p. Edge colors denote the connectivity such that grey is when dimensions of product spaces differ by 1, red is for $d_{GH}(\mathbb{E}^2, \mathbb{H}^2)$, blue is for $d_{GH}(\mathbb{S}^2, \mathbb{H}^2)$, and black is for $d_{GH}(\mathbb{E}^2, \mathbb{S}^2)$.

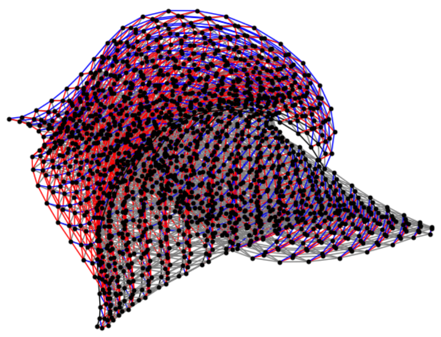

(a) $n_p = 4$.


(b) $n_p = 7$.


(c) $n_p = 10$.


(d) $n_p = 13$.


(e) $n_p = 16$.


(f) $n_p = 20$.

4 Experiments

We consider the task of image reconstruction on real-world datasets. To do this, we employ the autoencoder method described in Sáez de Ocáriz Borde et al. [2024] where the latent vector output of the encoder is projected onto the selected product space before reconstruction by the decoder. We evaluate the performance of a particular latent geometry by the reconstruction loss on the testing set after the autoencoder completes training.

We focus primarily on the MNIST and KMNIST datasets where we consider a search space of of $n_p = 5$ model spaces, each with dimensionality 2. This means the latent space has dimensionality 10.

4.1 Model Architecture

The autoencoder architecture used in our experiments consists of an encoder and a decoder. The encoder takes a 28x28 grayscale image as input and applies a series of convolutional layers, each followed by batch normalization and ReLU activation. For replicability, the specific architecture details are provided in 2.

The latent dimension (`latent_dim`) is determined by the number and dimensionality of the model spaces used in the product space. Before passing the latent representation to the decoder, it is projected onto the selected product space using its exponential map. This stereographic projection allows the latent space to inherit the geometric properties of the candidate manifold, enabling the exploration of different latent geometries during the search process.

Autoencoder Architecture Details

Encoder:

- Conv2d(1, 20, kernel_size=3, padding=1) - BatchNorm2d(20) - ReLU
- Conv2d(20, 20, kernel_size=3, padding=1) - BatchNorm2d(20) - ReLU
- Conv2d(20, 20, kernel_size=3, stride=2, padding=1) - BatchNorm2d(20) - ReLU
- Conv2d(20, 20, kernel_size=3, padding=1) - BatchNorm2d(20) - ReLU
- Conv2d(20, 20, kernel_size=3, padding=1) - BatchNorm2d(20) - ReLU
- Conv2d(20, 2, kernel_size=3, padding=1) - BatchNorm2d(2) - ReLU
- AdaptiveAvgPool2d(1)
- Flatten()

Decoder:

- Linear(2, 280) - ReLU
- Unflatten(1, (20, 7, 7))
- ConvTranspose2d(20, 20, kernel_size=3, stride=2, padding=1, output_padding=1) - ReLU - BatchNorm2d(20)
- ConvTranspose2d(20, 20, kernel_size=3, stride=2, padding=1, output_padding=1) - ReLU - BatchNorm2d(20)
- Conv2d(20, 1, kernel_size=3, padding=1) - Sigmoid

Geometric Autoencoder:

- Encoder
- ExponentialMap(product_manifold, latent_dim)
- Decoder

Figure 2: Detailed architecture of the geometric autoencoder used in experiments.

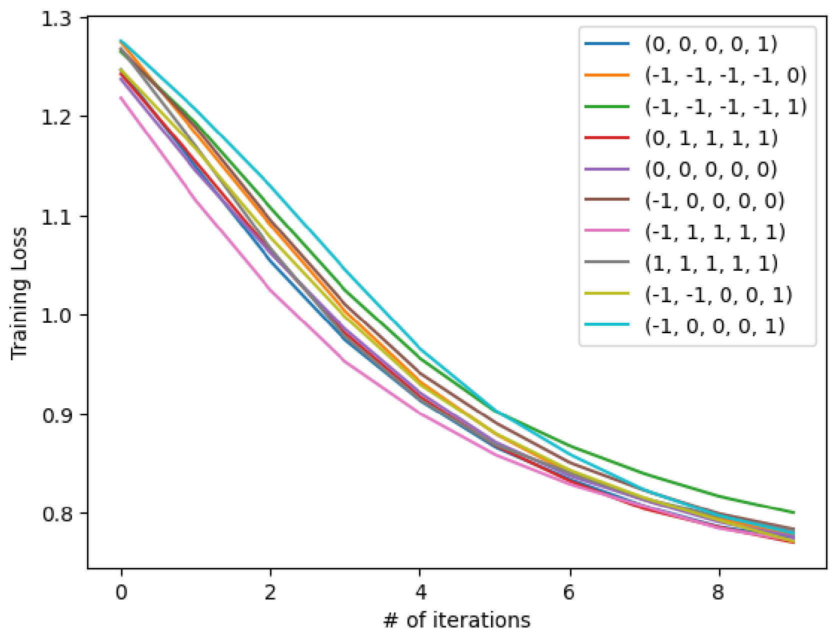


Figure 3: Plot of losses for different geometries using random search through the combinatorial space of product manifolds. We use $n_p = 5$, and each product manifold is represented as a tuple with each model space's curvature. For instance, $(-1, 0, 0, 0, 1)$ represents $\mathbb{H}^2 \times \mathbb{E}^2 \times \mathbb{E}^2 \times \mathbb{E}^2 \times \mathbb{S}^2$

.

4.2 MNIST

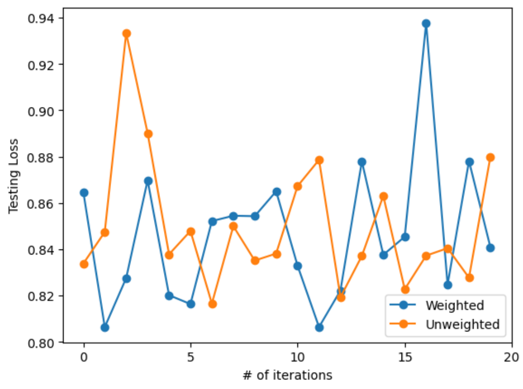
We conduct experiments on the MNIST dataset, which consists of 70,000 grayscale images of handwritten digits (60,000 for training and 10,000 for testing). The data preprocessing pipeline involves converting the images to tensors and rescaling the pixel values from the range [0, 255] to [0, 1]. Subsequently, the tensors are normalized by centering the data distribution around zero mean and scaling it to unit variance, using the mean and standard deviation values specific to the MNIST dataset (0.1307 and 0.3081, respectively). The entire training dataset and the full test dataset are used to create data loaders with a batch size of 64. The latent space dimensionality is determined by the number of component manifolds $n_p = 5$, resulting in a latent dimensionality of 10. The model is trained for 10 epochs with the Adam optimizer with a learning rate of 0.001 [Kingma and Ba, 2014]. Future work may consider approximating the performance metric on the validation set with speed of convergence of the model for the first few steps, which has been shown to be a reasonable indicator of performance [Smith, 2018].

Figure 3 presents the training loss trajectories for different latent geometries when trained on the MNIST dataset. The results confirm that the choice of latent geometry has a significant impact on the model's performance, as evidenced by the varying convergence patterns and final loss values. Notably, the trajectory corresponding to the geometry $\mathbb{H}^2 \times \mathbb{E}^2 \times \mathbb{E}^2 \times \mathbb{E}^2 \times \mathbb{E}^2$ converged to the lowest loss among the evaluated geometries, suggesting that this particular combination of hyperbolic and Euclidean spaces is well-suited for capturing the intrinsic structure of the MNIST dataset relative to the other product manifolds evaluated. We use reconstruction loss on the validation set for subsequent experiments.

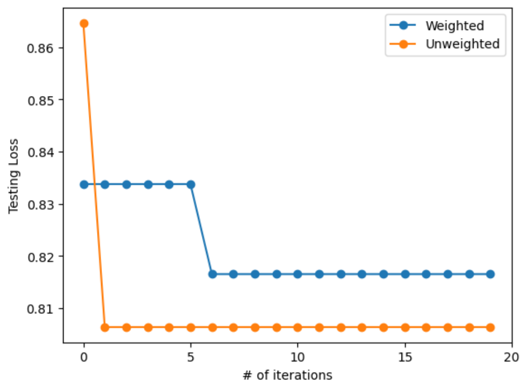
4.3 Unweighted vs Inverse d_{GH}-Weighted Random Walk

We compare the performance of unweighted and inverse d_{GH}-weighted random walks on the graph search space. In the unweighted case, the transition probabilities between product manifolds are uniform, meaning that the random walk is equally likely to step to any neighboring manifold, regardless of their geometric similarity. In contrast, the inverse d_{GH}-weighted random walk assigns higher transition probabilities to geometrically similar manifolds.

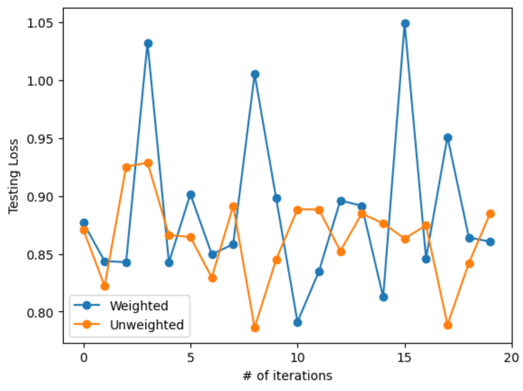
Figure 4: Experimental results: Comparison of losses over 20 iterations of random walk over graph for naive and backtracking searches.



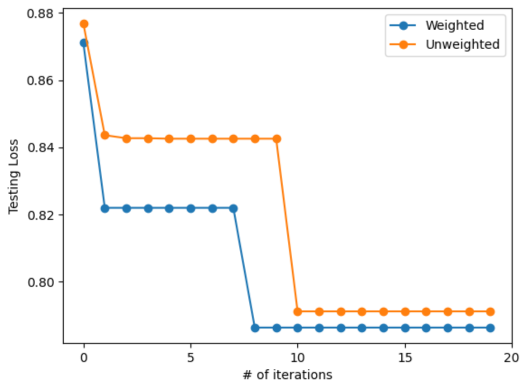
(a) All losses using naive random walk.



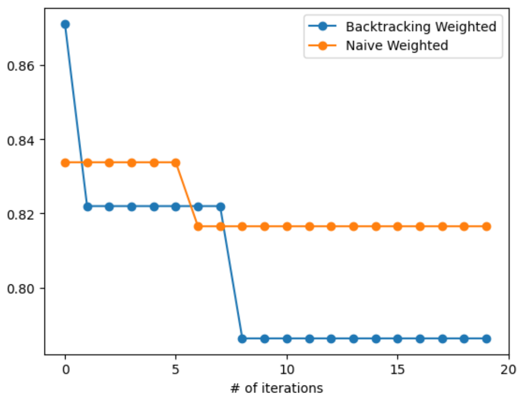
(b) Minimum losses using naive random walk.



(c) All losses using backtracking random walk.



(d) Minimum losses using backtracking random walk.



(e) Naive vs backtracking random walk.

From our experiments, we observed that the graph weighted using d_{GH} performed more favorably than the unweighted graph. This result points to the work done by Sáez de Ocáriz Borde et al. [2024] being valid for this scenario. We can also notice that in some of the iterations in Figure 4a, the unweighted version had lower losses compared to the weighted one. We hypothesize that it is due to the weighted approach getting stuck in a cluster of product manifolds with bad performance. Nonetheless, the minimum losses of weighted versus unweighted shows that there is a lot of potential for this method.

4.4 Backtracking vs Non-Backtracking Random Walk

In this subsection, we compare the performance of random walks with and without backtracking. As mentioned earlier, backtracking allows the random walk to revert to the previous product manifold if a step leads to a decrease in performance, mimicking the behavior of stochastic gradient descent. On the other hand, non-backtracking random walks continue exploring the graph search space without the ability to retrace their steps.

In Figure 4e, we can clearly see the backtracking approach performed better when we consider the minimum losses. As a result, it is very possible the graph search space may be monotically decreasing towards the minimum. With more iterations, it is possible for the backtracking approach to theoretically outperform the naive method even more.

5 Conclusion

In this project, we implemented the neural latent geometry search (NLGS) algorithm, as described by Sáez de Ocáriz Borde et al. [2024], and conducted experiments on MNIST. Our approach involved searching over a graph defined by product manifolds, projecting the latent vectors of autoencoders onto these manifolds, and minimizing the reconstruction loss using random walks.

The key findings of our experiments demonstrate that geometry-informed graphs, with edge weights based on the inverse Gromov-Hausdorff distance, generally perform similarly to unweighted graphs (though potentially limited by the available computational power). Furthermore, backtracking random walks consistently achieve superior performance compared to non-backtracking, naive random walks. These results highlight that there is potential importance in considering the geometric structure of the latent space when designing and optimizing neural networks.

However, the current NLGS framework has certain limitations. It is restricted to considering only product manifolds of constant curvature model spaces as the latent space manifolds. Additionally, the exponential growth of the graph search space, coupled with the expensive objective function evaluation, limits the method's applicability to small latent space dimensions. The extra performance gains may not justify the significant increase in time and complexity of the method, indicating that it is not yet ready for practical use.

To address these limitations and further advance the field of NLGS, several research directions can be explored. Investigating the impact of enforcing latent geometries on different tasks may lead to improved representations across a wide range of applications. Future work should focus on alleviating the combinatorial explosion of the graph search space or developing principled heuristics for identifying fruitful regions within it. Approximating the performance metric with heuristics such as the speed of initial convergence could also be considered. Moreover, enhancing the graph search process with geometrically-aware optimization algorithms is another promising avenue for future research.

Our work provides an initial attempt to understand the geometry of the space of product manifolds from a spectral clustering perspective through the use of random walks. We applied this method to improve the performance of autoencoders and introduced and experimented with different random walk techniques on the graph search space. While this approach requires further development before it becomes practically useful, it represents a solid foundation for an intriguing endeavor to enhance deep learning in a theoretically principled manner.

References

S.-i. Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5): 185–196, 1993.

S. Andradóttir. A method for discrete stochastic optimization. *Management Science*, 41(12):1946–1961, 1995.

L. J. Billera, S. P. Holmes, and K. Vogtmann. Geometry of the space of phylogenetic trees. *Advances in Applied Mathematics*, 27(4):733–767, 2001.

A. M. Bronstein, M. M. Bronstein, and R. Kimmel. Generalized multidimensional scaling: a framework for isometry-invariant partial surface matching. *Proceedings of the National Academy of Sciences*, 103(5):1168–1172, 2006.

M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

C. Chadebec and S. Allassonnière. A geometric perspective on variational autoencoders. *Advances in Neural Information Processing Systems*, 35:19618–19630, 2022.

H. S. de Ocariz Borde, A. Arroyo, I. Morales, I. Posner, and X. Dong. Gromov-hausdorff distances for comparing product manifolds of model spaces, 2023.

L. Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012. doi: 10.1109/MSP.2012.2211477.

C. Ding, X. He, H. Zha, M. Gu, and H. Simon. Spectral min-max cut for graph partitioning and data clustering. 2001.

Y. Fei, X. Wei, Y. Liu, Z. Li, and M. Chen. A survey of geometric optimization for deep learning: From euclidean space to riemannian manifold. *arXiv preprint arXiv:2302.08210*, 2023.

P. I. Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

F. Göbel and A. Jagers. Random walks on graphs. *Stochastic processes and their applications*, 2(4): 311–336, 1974.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

W. L. Hamilton, R. Ying, and J. Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

S. Hauberg, O. Freifeld, and M. Black. A geometric take on metric learning. *Advances in Neural Information Processing Systems*, 25, 2012.

G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

P. Jawanpuria, A. Balgovind, A. Kunchukuttan, and B. Mishra. Learning multilingual word embeddings in latent metric space: a geometric approach. *Transactions of the Association for Computational Linguistics*, 7:107–120, 2019.

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

J. M. Lee and J. M. Lee. *Smooth manifolds*. Springer, 2012.

M. Nickel and D. Kiela. Poincaré embeddings for learning hierarchical representations. *CoRR*, abs/1705.08039, 2017. URL `http://arxiv.org/abs/1705.08039`.

D. C. Plaut and G. E. Hinton. Learning sets of filters using back-propagation. *Computer Speech & Language*, 2(1):35–61, 1987.

R. Rabadán and A. J. Blumberg. *Topological data analysis for genomics and evolution: topology in biology*. Cambridge University Press, 2019.

H. Sáez de Ocáriz Borde, A. Arroyo, I. Morales, I. Posner, and X. Dong. Neural latent geometry search: Product manifold inference via gromov-hausdorff-informed bayesian optimization. *Advances in Neural Information Processing Systems*, 36, 2024.

A. Shukla, S. Uppal, S. Bhagat, S. Anand, and P. Turaga. Geometry of deep generative models for disentangled representations. In *Proceedings of the 11th Indian Conference on Computer Vision, Graphics and Image Processing*, pages 1–8, 2018.

L. N. Smith. A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*, 2018.

M. Stephan, M. D. Hoffman, D. M. Blei, et al. Stochastic gradient descent as approximate bayesian inference. *Journal of Machine Learning Research*, 18(134):1–35, 2017.

U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17:395–416, 2007.