Convolutional Neural Networks on Manifolds: From Graphs and Back

Zhiyang Wang Luana Ruiz Alejandro Ribeiro

***Abstract*—Geometric deep learning has gained much attention in recent years due to more available data acquired from non-Euclidean domains. Some examples include point clouds for 3D models and wireless sensor networks in communications. Graphs are common models to connect these discrete data points and capture the underlying geometric structure. With the large amount of these geometric data, graphs with arbitrarily large size tend to converge to a limit model – the manifold. Deep neural network architectures have been proved as a powerful technique to solve problems based on these data residing on the manifold. In this paper, we propose a manifold neural network (MNN) composed of a bank of manifold convolutional filters and point-wise nonlinearities. We define a manifold convolution operation which is consistent with the discrete graph convolution by discretizing in both space and time domains. To sum up, we focus on the manifold model as the limit of large graphs and construct MNNs, while we can still bring back graph neural networks by the discretization of MNNs. We carry out experiments based on point-cloud dataset to showcase the performance of our proposed MNNs.**

***Index Terms*—Manifold convolution, manifold neural networks, geometric deep learning**

I. Introduction

Convolutional neural networks (CNNs) have achieved impressive success in a wide range of applications, including but not limited to natural language processing [1], image denoising [2] and video analysis [3]. Convolution operations are implemented to capture the local information and features based on the characteristics of the dataset. The remarkable success provides the support that CNNs are recognized as powerful techniques when processing traditional signals such as sound, image or video, which all lie in the Euclidean domains. As we have more access to larger scale data and stronger computing power, increasing attention is being paid to processing data lying in the non-Euclidean domains.

Many practical problems rely on non-Euclidean data. There is the case, for example, detection and recommendation in social networks [4], resource allocations over wireless networks [5], point clouds for shape segmentation [6]. There have been works that extend the CNN architecture to non-Euclidean domains [7]–[9], which reproduce the success of CNNs in Euclidean domains. Among these models, graphs are commonly used to construct the underlying data structure, while the graph size scales with the amount of data. In this work, we aim to construct CNNs on a more general non-Euclidean domain – the manifold. Graphs with well-defined limits are shown to converge to a manifold model [10], [11], which makes the manifold neural networks capable of capturing properties for a series of graphs. The convolution operation is not taken for granted in non-Euclidean domains due to the lack of global parametrization and shift invariance. We define a manifold convolution operation based on the heat diffusion process controlled by the Laplace-Beltrami operator. We construct a manifold convolutional filter to process manifold signals. By cascading the layers consisting of manifold filter banks and nonlinear activation functions, we can define the manifold neural networks (MNNs) as a deep learning framework on the manifold. To motivate the practical implementations of our proposed MNNs, we first discretize the MNN in the space domain by sampling points on the manifold. The proposed MNN can be transferred to this discretized manifold as a discretized MNN which converges to the underlying MNN. We further carry out discretization in the time domain by sampling the filter impluse function in discrete and finite time steps. In this way, we can not only execute our proposed MNNs, but also recover the graph convolutions and graph neural networks [7]. This concludes our thought starting from a graph sequence to the limit as a manifold and back to the graphs. We finally verify the performance of our proposed MNN with a point cloud based model classification problem.

Related works include neural networks built on graphons [12], [13], which are limits of a sequence of graphs. Different from manifolds, graphons only represent the limits for graphs with unbounded degrees [14]. Stability of MNNs have been studied considering the perturbations to the Laplace-Beltrami operator [9], [15]. In this paper we focus more on the establishment of manifold convolutions and how graph convolutions are recovered by discretization. A general framework for algebraic neural networks has been proposed for architectures unified with commutative algebras [16].

The rest of the paper is organized as follows. We start with some preliminary concepts and define the manifold convolutions in Section II. We construct the MNNs based on manifold filters in Section III. In Section IV, we implement the discretization in space and time domains to make the MNNs realizable which also bring back to graph convolutions. Our proposed MNN is verified in a model classification problem in Section V. The conclusions are presented in Section VI.

II. Manifold Convolution

A. Preliminary Definitions

In this paper, we consider a compact, smooth, and differentiable d-dimensional submanifold $\mathcal{M}$ embedded in $\mathbb{R}^N$. The

Supported by NSF CCF 1717120, Theorinet Simons and ARL DCIST CRA under Grant W911NF-17-2-0181. The authors are with Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, Pennsylvania, USA

embedding induces a Riemannian structure [17] on $\mathcal{M}$ which endows a measure μ over the manifold. *Manifold signals* supported on $\mathcal{M}$ are smooth scalar functions $f : \mathcal{M} \to \mathbb{R}$. We consider manifold signals in a Hilbert space in which we define the inner product as

$$\langle f, g \rangle_{L^2(\mathcal{M})} = \int_{\mathcal{M}} f(x) g(x) \mathrm{d}\mu(x) \tag{1}$$

with repsect to the measure μ.

The manifold is locally Euclidean, which elicits *intrinsic gradient* for differentiation as a local operator [11]. The local Euclidean space around $x \in \mathcal{M}$ containing all of the vectors tangent to $\mathcal{M}$ at x is denoted as tangent space $T_x\mathcal{M}$. We use $T\mathcal{M}$ to represent the disjoint union of all tangent spaces on $\mathcal{M}$. The intrinsic gradient can thus be written as an operator $\nabla : L^2(\mathcal{M}) \to L^2(T\mathcal{M})$ mapping scalar functions to tangent vector functions on $\mathcal{M}$. The adjoint operator of intrinsic gradient is the *intrinsic divergence* defined as $\mathrm{div} : L^2(T\mathcal{M}) \to L^2(\mathcal{M})$. Based on these two differentiation operators, the Laplace-Beltrami (LB) operator $\mathcal{L} : L^2(\mathcal{M}) \to L^2(\mathcal{M})$ can be defined as the intrinsic divergence of the intrinsic gradient [18], formally as

$$\mathcal{L} f = -\mathrm{div} \circ \nabla f = -\nabla \cdot \nabla f. \tag{2}$$

Similar to the Laplacian operator in Euclidean domains or the Laplace matrix in graphs [19], the LB operator evaluates how much the function value at point x differs from the average function value of its neighborhood [11].

The LB operator provides a basis for expressing and solving physical tasks by Partial Differential Equations (PDEs). One of the remarkable applications is characterizing the heat diffusion over manifolds by the *heat equation*

$$\frac{\partial u(x,t)}{\partial t} + \mathcal{L} u(x,t) = 0, \tag{3}$$

where $u(x,t) \in L^2(\mathcal{M})$ measures the temperature at $x \in \mathcal{M}$ at time $t \in \mathbb{R}^+$. With initial condition given by $u(x,0) = f(x)$, the solution can be expressed as

$$u(x,t) = e^{-t\mathcal{L}} f(x), \tag{4}$$

which provides an essential element to construct manifold convolution in the following.

Due to the compactness of $\mathcal{M}$, the LB operator $\mathcal{L}$ is self-adjoint and positive-semidefinite. This means that $\mathcal{L}$ possesses a real positive spectrum $\{\lambda_i\}_{i=1}^{\infty}$ with the eigenvalues λ_i and the corresponding eigenfunctions ϕ_i satisfying

$$\mathcal{L} \boldsymbol{\phi}_i = \lambda_i \boldsymbol{\phi}_i \tag{5}$$

The eigenvalues are ordered in an increasing order as $0 < \lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \ldots$. According to Weyl's law [20], we have $\lambda_i \propto i^{2/d}$ for a d-dimensional manifold. The orthonormal eigenfunctions ϕ_i form a general eigenbasis of $L^2(\mathcal{M})$ in the intrinsic sense.

Since $\mathcal{L}$ is a total variation operator, the eigenvalues λ_i can be interpreted as the canonical frequencies and the eigenfunctions $\boldsymbol{\phi}_i$ as the canonical oscillation modes of $\mathcal{M}$.

B. Manifold Convolutional Filters

Convolution operation is a powerful technique to give zero state response to any input signal with the filter impulse response of the system [21]. Similar to time signals are processed with time convolutions and graph signals are processed by graph convolutions [22], we define manifold convolution with a filter impulse response $\tilde{h}$ and manifold signal f.

Definition 1 (Manifold filter) *Let $\tilde{h} : \mathbb{R}^+ \to \mathbb{R}$ and let $f \in L^2(\mathcal{M})$ be a manifold signal. The manifold filter with impulse response $\tilde{h}$, denoted $\mathbf{h}$, is given by*

$$g(x) = (\mathbf{h} f)(x) := \int_0^{\infty} \tilde{h}(t) u(x,t) dt \tag{6}$$

where $u(x,t)$ is the solution of the heat equation (3) *with $u(x,0) = f(x)$. Substitute the solution $u(x,t)$ with* (4)*, and we can derive a parametric form of $\mathbf{h}$ as*

$$g(x) = (\mathbf{h} f)(x) = \int_0^{\infty} \tilde{h}(t) e^{-t\mathcal{L}} f(x) dt = \mathbf{h}(\mathcal{L}) f(x). \tag{7}$$

Manifold filters are local spatial operators operating directly on points on the manifold based on the LB operator. The exponential term $e^{-t\mathcal{L}}$ can be interpreted as a shift operator like the time delay in a Linear-Time Invariant (LTI) filter [21] and the graph shift in a Linear-Shift Invariant (LSI) graph filter [22]. In fact, manifold filters can recover graph filters by discretizing in both space and time domains, which we discuss thoroughly in Section IV.

The LB operator $\mathcal{L}$ possesses the eigendecomposition $\{\lambda_i, \phi_i\}_{i=1}^{\infty}$. Eigenvalue λ_i can be interpreted as the canonical frequency and the eigenfunction ϕ_i as the canonical oscillation mode. By projecting a manifold signal f onto the eigenfunction, we can write the *frequency representation* $\hat{f}$ as

$$[\hat{f}]_i = \langle f, \boldsymbol{\phi}_i \rangle_{L^2(\mathcal{M})} = \int_{\mathcal{M}} f(x) \boldsymbol{\phi}_i(x) \mathrm{d}\mu(x). \tag{8}$$

The spectrum and eigenbasis of the LB opertor help to understand the frequency behavior of the manifold filter $\mathbf{h}(\mathcal{L})$. The frequency representation of manifold filter output g can be similarly written as

$$[\hat{g}]_i = \int_{\mathcal{M}} \int_0^{\infty} \tilde{h}(t) e^{-t\mathcal{L}} f(x) \mathrm{d}t \boldsymbol{\phi}_i(x) \mathrm{d}\mu(x). \tag{9}$$

By substituting $e^{-t\mathcal{L}} \phi_i = e^{-t\lambda_i} \phi_i$, we can get

$$[\hat{g}]_i = \int_0^{\infty} \tilde{h}(t) e^{-t\lambda_i} \mathrm{d}t [\hat{f}]_i. \tag{10}$$

The function solely dependent on λ_i is defined as the *frequency response* of the filter $\mathbf{h}(\mathcal{L})$.

Definition 2 (Frequency response) *The frequency response of the filter $\mathbf{h}(\mathcal{L})$ is given by*

$$\hat{h}(\lambda) = \int_0^{\infty} \tilde{h}(t) e^{-t\lambda} dt, \tag{11}$$

which leads (10) *to $[\hat{g}]_i = \hat{h}(\lambda_i)[\hat{f}]_i$.*

Definition 2 indicates that the frequency response of manifold filter is point-wise in frequency domain. Combining the frequency representation of $\hat{g}$ over the whole spectrum, we can obtain the frequency representation of manifold filter $\mathbf{h}$ as

$$g = \mathbf{h}(\mathcal{L})f = \sum_{i=1}^{\infty} \hat{h}(\lambda_i)\langle f, \phi_i\rangle_{L^2(\mathcal{M})}\phi_i. \tag{12}$$

The well-defined manifold filters make an important building block in Manifold Neural Networks (MNNs), which we show in the following section.

III. Manifold Neural Networks

Manifold neural networks (MNNs) augment manifold filters with a point-wise nonlinear activation function. We extend the definition to a mapping on the manifold $\sigma : L^2(\mathcal{M}) \to L^2(\mathcal{M})$ as an independent application on each point of the manifold. In a single-layer MNN, the manifold signal f is passed through a manifold filter followed by a point-wise nonlinearity as

$$y(x) = \sigma\left(\mathbf{h}(\mathcal{L})f(x)\right), \tag{13}$$

which can be seen as a basic nonlinear processing of the input manifold signal. By stacking this procedure in layers, a multilayer MNN can be constructed which can be formally written as a function composition. The output manifold signal of a layer becomes the input signal of the next layer. Let $l = 1, 2, \cdots, L$ stand for the index for the layer and $\mathbf{h}_l(\mathcal{L})$ as the manifold filter on each layer. For a specific layer l, filter $\mathbf{h}_l(\mathcal{L})$ takes the output $f_{l-1}(x)$ as the input produces the output of layer l as

$$f_l(x) = \sigma\left(\mathbf{h}_l(\mathcal{L})f_{l-1}(x)\right), \tag{14}$$

where $f_0(x) = f(x)$ as the given input manifold signal. After a recursive applications through L layers, we can get the output of the MNN as $f_L(x)$.

When considering multiple features in each layer to increase the representation power of MNN, the manifold filters map the input F_{l-1} features from layer $l-1$ to F_l intermediate features in layer l with a bank of manifold filters, i.e.,

$$y_l^p(x) = \sum_{q=1}^{F_{l-1}} \mathbf{h}_l^{pq}(\mathcal{L})f_{l-1}^q(x), \tag{15}$$

where $\mathbf{h}_l^{pq}(\mathcal{L})$ is the filter mapping the q-th feature from layer $l-1$ to the p-th feature of layer l, for $1 \leq q \leq F_{l-1}$ and $1 \leq p \leq F_l$. The intermediate features are then processed by the nonlinearity σ as

$$f_l^p(x) = \sigma\left(y_l^p(x)\right). \tag{16}$$

The output of layer L is the output of MNN with F_L features. To represent the MNN more precisely, we gather the impulse responses of all the manifold filters $\mathbf{h}_l^{pq}$ as a function set $\mathbf{H}$ and define the MNN as a map $\mathbf{\Phi}(\mathbf{H}, \mathcal{L}, f)$. This map is parameterized by both the filter functions $\mathbf{H}$ and the LB operator $\mathcal{L}$.

IV. Discretization in Space and Time

MNNs are built based on manifold convolutional filters (Definition 1) processing manifold signals over an infinite time horizon. In practice, the continuous architectures cannot be implemented directly. In this section, we discuss the practical application of MNNs (16) by discretization in both space and time domains.

A. Discretization in the Space Domain

Realistically, the underlying manifold along with its LB operator is inaccessible directly. It is common to use sampling points to form a point cloud as an approximation of the manifold structure.

With the knowledge of the coordinates of the sampling points, the underlying manifold structure can be approximated by a geometric graph structure which can also be seen as a discretized manifold [10], [23]. The graph Laplacian is hence the approximation of the LB operator, whose convergence to the LB operator as the number of sampling points increases has been shown explicitly [23], [24].

Specifically, we model the set of n sampling points as $X = \{x_1, x_2, \ldots, x_n\}$ which are sampled i.i.d. from measure μ of manifold $\mathcal{M} \subset \mathbb{R}^N$. A complete weighted symmetric graph $\mathbf{G}_n$ can be constructed by seeing the sampling points as the vertices of the graph and the Euclidean distance between pairs of points as the edge weights. To be more precise, the edge weight w_{ij} connecting point x_i and x_j is given by

$$w_{ij} = \exp\left(-\frac{\|x_i - x_j\|^2}{4t_n}\right), \tag{17}$$

with $\|x_i - x_j\|$ representing the Euclidean distance between x_i and x_j. Parameter t_n controls the chosen Gaussian kernel [10]. The adjacency matrix of $\mathbf{G}_n$ is thus defined as $[\mathbf{A}_n]_{ij} = w_{ij}$ for $1 \leq i, j \leq n$ with $\mathbf{A}_n \in \mathbb{R}^{n\times n}$. The correspondent graph Laplacian matrix $\mathbf{L}_n$ [25] thus can be defined as

$$\mathbf{L}_n = \text{diag}(\mathbf{A}_n\mathbf{1}) - \mathbf{A}_n, \tag{18}$$

which is the approximated LB operator of the discretized manifold $\mathbf{L}_n$. We define a uniform sampling operator $\mathbf{P}_n : L^2(\mathcal{M}) \to L^2(X)$ to discretize the manifold signal f as a graph signal $\mathbf{x}_n \in \mathbb{R}^N$, which is defined as

$$\mathbf{x}_n = \mathbf{P}_n f \text{ with } [\mathbf{x}_n]_i = f(x_i), \quad x_i \in X, \tag{19}$$

which indicates that the i-th element of the discretized graph signal $\mathbf{x}_n$ is the evaluation of manifold signal f at the sampling point x_i.

The definition of manifold filter (Definition 1) has shown that it can be parameterized by the LB operator. Likewise, we can replace the LB operator with the discrete Laplacian operator and turns $\mathbf{h}$ to a discrete manifold filter, i.e.,

$$\mathbf{z}_n = \int_0^{\infty} \tilde{h}(t)e^{-t\mathbf{L}_n}\mathrm{d}t\mathbf{x}_n = \mathbf{h}(\mathbf{L}_n)\mathbf{x}_n, \qquad \mathbf{x}_n, \mathbf{z}_n \in \mathbb{R}^n, \tag{20}$$

where $\mathbf{z}_n$ is the output discrete graph signal. By cascading the layers containing discrete manifold filters and point-wise nonlinearities, we can construct a neural network on this discreteized manifold as

$$\mathbf{x}_l^p = \sigma\left(\sum_{q=1}^{F_{l-1}} \mathbf{h}_l^{pq}(\mathbf{L}_n)\mathbf{x}_{l-1}^q\right), \tag{21}$$

where $\mathbf{h}_l^{pq}$ in the filter bank maps the features in the $l-1$-th layer to the p-th feature in the l-th layer. Each layer contains F_l features. By gathering all the filter impulse functions as a set $\mathbf{H}$, we can represent the neural network on the discrete manifold as a map $\mathbf{\Phi}(\mathbf{H}, \mathbf{L}_n, \mathbf{x})$.

As the number of sampling points goes to infinity, the discrete graph signal $\mathbf{x}_n$ converges to the manifold signal f while the discrete graph Laplacian matrix $\mathbf{L}_n$ also converges to the LB operator $\mathcal{L}$ of the underlying manifold [10]. Combining these convergence, it is reasonable to conclude that the output of the neural network on the discrete manifold converges to the output of the continuous manifold neural network, which we state explicitly as the following proposition.

Proposition 1 *Let $X = \{x_1, x_2, ...x_n\}$ be a set of n sampling points i.i.d. from manifold signal f of manifold $\mathcal{M} \subset \mathbb{R}^N$, sampled by an operator $\mathbf{P}_n$ (19). Let $\mathbf{G}_n$ be a discrete approximation of $\mathcal{M}$ constructed from X as in (17) with $t_n = n^{-1/(d+2+\alpha)}$ and $\alpha > 0$. Let $\mathbf{\Phi}(\mathbf{H}, \cdot, f)$ be a neural network parameterized by the LB operator $\mathcal{L}$ of the manifold $\mathcal{M}$ (16) or the discrete Laplacian operator $\mathbf{L}_n$ of the discretized manifold $\mathbf{G}_n$. It holds that*

$$\lim_{n\to\infty} \|\mathbf{\Phi}(\mathbf{H}, \mathbf{L}_n, \mathbf{P}_n f) - \mathbf{P}_n \mathbf{\Phi}(\mathbf{H}, \mathcal{L}, f)\| = 0, \tag{22}$$

with a probability one.

Proof. See Appendix in [26]. ■

Proposition 1 claims that neural networks constructed from the discrete Laplacian $\mathbf{L}_n$ can give a good approximation of the manifold neural networks.

B. Discretization in the Time Domain

Definition 1 shows that the manifold filter $\mathbf{h}$ is determined by the impulse response function $\tilde{h}(t)$. Therefore, if we want to learn the MNN, we need to learn function $\tilde{h}(t)$, which is infeasible without enough prior knowledge. To make this learning practical, we discretize function $\tilde{h}(t)$ in the continuous time domain with a fixed sampling interval T_s. We replace the filter response function with a series of coefficients $h_k = \tilde{h}(kT_s)$, $k = 0, 1, 2 \ldots$. For simplicity, we settle the sampling interval as a unit $T_s = 1$. The manifold convolution in the discrete time domain can be written as

$$\mathbf{h}(\mathcal{L})f(x) = \sum_{k=0}^{\infty} h_k e^{-k\mathcal{L}} f(x), \tag{23}$$

where $\{h_k\}_{k=0}^{\infty}$ are called filter coefficients or taps.

Considering the infinite number of the filter coefficients, the realization of $\mathbf{h}(\mathcal{L})f(x)$ is still impractical. We fix a time horizon of K samples over the time horizon and reformulate (24) as

$$\mathbf{h}(\mathcal{L})f(x) = \sum_{k=0}^{K-1} h_k e^{-k\mathcal{L}} f(x), \tag{24}$$

which corresponds to the form of a finite impulse response (FIR) filter with shift operator $e^{-\mathcal{L}}$. Similarly, we can discretize the filter on the discretized manifold in (20) over the time domain, which leads to a practical manifold filter on a discretized manifold and in the discrete time domain, i.e.,

$$\mathbf{z} = \mathbf{h}(\mathbf{L}_n)\mathbf{x} = \sum_{k=0}^{K-1} h_k e^{-k\mathbf{L}_n}\mathbf{x}. \tag{25}$$

We can observe that this discretized manifold filter recovers the form of graph convolution [22] with $e^{-\mathbf{L}_n}$ seen as the graph shift operator. By replacing the filter $\mathbf{h}_l^{pq}(\mathbf{L}_n)$ in (21) with (25), we further recover the graph neural network (GNN) architecture from MNNs. Till now we have completed the process of building MNNs from graphs and back. Manifolds can be seen as the limits of graphs and thus a tool for analyzing large graphs. CNNs can be constructed on manifolds to address problems in the limit sense. By further discretizing the constructed MNNs, we can bring the problem back to the graphs.

V. Simulations

We evaluate the performance of our proposed MNN structure with practical implementations in (21) and (25), i.e., we approximate the unrealizable MNN with discrete GNNs. We use the ModelNet10 dataset [27] and carry out a classification problem. The dataset contains 3,991 meshed CAD models from 10 categories for training and 908 models for testing. We sample 300 points from each meshed model uniformly and construct a point cloud with these 300 points. Explicitly, we see each point as a node and the Euclidean distance between each two points as the edge weight. Our goal is to identify the models for chair from other models as illustrated in Figure 1.

The point cloud models are here approximated by the dense graphs. The edge weights are calculated according to (17) with t_n set as 0.3. The Laplacian matrix is calculated with (18) for each point cloud model. We implement several different architectures to solve the classification problem, including graph filters (GF) and graph neural networks (GNN) with 1 and 2 layers respectively. The single layer architectures contain $F_0 = 3$ input features which are the coordinates of every point in 3d space. The output features are set as $F_1 = 64$. The architectures with 2 layers include another layer with $F_2 = 32$ features. The filter taps in each layer are set as $K = 5$. The nonlinearity is a ReLu function. All the architectures are concluded with a linear readout layer mapping the output features to a binary scalar to estimate the classifications.

We train all the architectures with an ADAM optimizer [28] with the learning rate as 0.005 and decaying factor as 0.9,0.999 to minimize the entropy loss. The training model set is divided into batches with 10 models over 40 epochs. We repeat 5 sampling realizations for all the architectures and calculate the average classification error rates as well as the standard

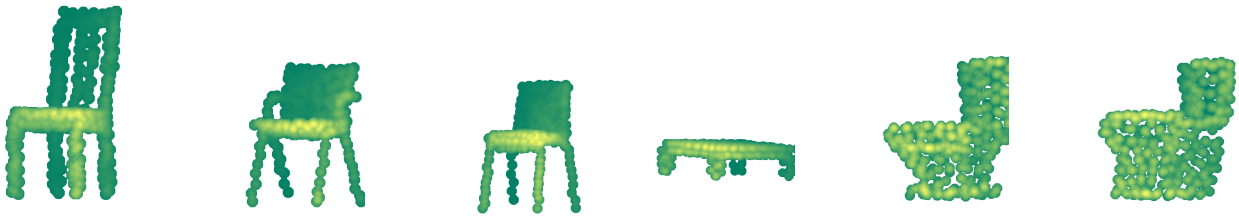

Fig. 1: Point cloud models with 300 sampling points in each model

deviation in Table I. We observe that graph neural networks perform better than the graph filters while architectures with more layers learn more accurate models with more parameters learned.

Architecture	error rates
GNN1Ly	$8.04\% \pm 0.88\%$
GNN2Ly	$4.30\% \pm 2.64\%$
GF1Ly	$13.77\% \pm 6.87\%$
GF2Ly	$12.22\% \pm 7.89\%$

TABLE I: Classification error rates for model 'chair' in the test dataset. Average over 5 data realizations. The number of nodes is $N = 300$.

VI. Conclusion

In this paper, we have studied the limit of a graph sequence as a manifold. We have defined a manifold convolution operation with the heat diffusion controlled by the Laplace-Beltrami operator. We have further constructed a manifold neural network architecture. To realize the MNN in practice, we have carried out discretization in both space and time domains which recovers the convolution and neural networks on graphs. We finally verified the performance of MNN with a model classification problem.

References

[1] W. Wang and J. Gang, "Application of convolutional neural network in natural language processing," in *2018 International Conference on Information Systems and Computer Aided Education (ICISCAE)*. IEEE, 2018, pp. 64–70.

[2] K. Zhang, W. Zuo, and L. Zhang, "Ffdnet: Toward a fast and flexible solution for cnn-based image denoising," *IEEE Transactions on Image Processing*, vol. 27, no. 9, pp. 4608–4622, 2018.

[3] K. Xu, L. Wen, G. Li, L. Bo, and Q. Huang, "Spatiotemporal cnn for video object segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1379–1388.

[4] M. Aggarwal and M. N. Murty, *Machine Learning in Social Networks: Embedding Nodes, Edges, Communities, and Graphs*. Springer Nature, 2020.

[5] Z. Wang, M. Eisen, and A. Ribeiro, "Learning decentralized wireless resource allocations with graph neural networks," *IEEE Transactions on Signal Processing*, vol. 70, pp. 1850–1863, 2022.

[6] Y. Xie, J. Tian, and X. X. Zhu, "Linking points with labels in 3d: A review of point cloud semantic segmentation," *IEEE Geoscience and Remote Sensing Magazine*, vol. 8, no. 4, pp. 38–59, 2020.

[7] F. Gama, A. G. Marques, G. Leus, and A. Ribeiro, "Convolutional neural network architectures for signals supported on graphs," *IEEE Transactions on Signal Processing*, vol. 67, no. 4, pp. 1034–1049, 2019.

[8] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *arXiv preprint arXiv:1606.09375*, 2016.

[9] Z. Wang, L. Ruiz, and A. Ribeiro, "Stability of neural networks on riemannian manifolds," in *2021 29th European Signal Processing Conference (EUSIPCO)*. IEEE, 2021, pp. 1845–1849.

[10] M. Belkin and P. Niyogi, "Towards a theoretical foundation for laplacian-based manifold methods," *Journal of Computer and System Sciences*, vol. 74, no. 8, pp. 1289–1308, 2008.

[11] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[12] L. Ruiz, L. Chamon, and A. Ribeiro, "Graphon neural networks and the transferability of graph neural networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1702–1712, 2020.

[13] L. Ruiz, L. F. Chamon, and A. Ribeiro, "Graphon signal processing," *IEEE Transactions on Signal Processing*, vol. 69, pp. 4961–4976, 2021.

[14] L. Lovász, *Large networks and graph limits*. American Mathematical Soc., 2012, vol. 60.

[15] Z. Wang, L. Ruiz, and A. Ribeiro, "Stability of neural networks on manifolds to relative perturbations," *arXiv preprint arXiv:2110.04702*, 2021.

[16] A. Parada-Mayorga and A. Ribeiro, "Algebraic neural networks: Stability to deformations," *IEEE Transactions on Signal Processing*, vol. 69, pp. 3351–3366, 2021.

[17] J. Gallier and J. Quaintance, *Differential geometry and Lie groups: a computational perspective*. Springer Nature, 2020, vol. 12.

[18] S. Rosenberg and R. Steven, *The Laplacian on a Riemannian manifold: an introduction to analysis on manifolds*. Cambridge University Press, 1997, no. 31.

[19] P. Moon and D. E. Spencer, *Field theory handbook: including coordinate systems, differential equations and their solutions*. Springer, 2012.

[20] W. Arendt, R. Nittka, W. Peter, and F. Steiner, "Weyl's law: Spectral properties of the laplacian in mathematics and physics," *Mathematical analysis of evolution, information, and complexity*, pp. 1–71, 2009.

[21] A. V. Oppenheim, A. S. Willsky, S. H. Nawab, G. M. Hernández *et al.*, *Signals & systems*. Pearson Educación, 1997.

[22] F. Gama, E. Isufi, G. Leus, and A. Ribeiro, "Graphs, convolutions, and neural networks: From graph filters to graph neural networks," *IEEE Signal Processing Magazine*, vol. 37, no. 6, pp. 128–138, 2020.

[23] D. B. Dunson, H.-T. Wu, and N. Wu, "Spectral convergence of graph laplacian and heat kernel reconstruction in l from random samples," *Applied and Computational Harmonic Analysis*, vol. 55, pp. 282–336, 2021.

[24] J. Calder and N. G. Trillos, "Improved spectral convergence rates for graph laplacians on epsilon-graphs and k-nn graphs," *arXiv preprint arXiv:1910.13476*, 2019.

[25] R. Merris, "A survey of graph laplacians," *Linear and Multilinear Algebra*, vol. 39, no. 1-2, pp. 19–31, 1995.

[26] "Convolutional neural networks on manifolds: From graphs and back." [Online]. Available: https://zhiyangw.com/Papers/convolution-asilomar2022.pdf

[27] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1912–1920.

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

[29] M. Belkin and P. Niyogi, "Convergence of laplacian eigenmaps," *Advances in neural information processing systems*, vol. 19, 2006.

APPENDIX

A. Proof of Proposition 1

We first import the existing results from [29] which indicates the spectral convergence of the constructed Laplacian operator based on the discretized manifold to the LB operator of the underlying manifold.

Theorem 1 (Theorem 2.1 [29]) *Let λ_i^n be the i-th eigenvalue of $\mathbf{L}_n$ and $\boldsymbol{\phi}_i^n$ be the corresponding eigenfunction. Let λ_i and $\boldsymbol{\phi}_i$ be the corresponding eigenvalue and eigenfunction of $\mathcal{L}$ respectively. Then there exists a sequence $t_n \to 0$, such that*

$$\lim_{n\to\infty} \lambda_i^n = \lambda_i, \qquad \lim_{n\to\infty} \|\boldsymbol{\phi}_i^n - \boldsymbol{\phi}_i\| = 0, \tag{26}$$

where the limits are taken in probability.

With the definitions of neural networks on discretized manifold and manifold, the output difference can be written as

$$\begin{aligned}\|\boldsymbol{\Phi}(\mathbf{H}, \mathbf{L}_n, \mathbf{P}_n f) - \mathbf{P}_n \boldsymbol{\Phi}(\mathbf{H}, \mathcal{L}, f))\| &= \left\| \sum_{q=1}^{F_L} \mathbf{x}_L^q - \sum_{q=1}^{F_L} \mathbf{P}_n f_L^q \right\| \\ &\leq \sum_{q=1}^{F_L} \|\mathbf{x}_L^q - \mathbf{P}_n f_L^q\|.\end{aligned} \tag{27}$$

By inserting the definitions, we have

$$\begin{aligned}&\|\mathbf{x}_l^p - \mathbf{P}_n f_l^p\| \\ &= \left\| \sigma\left(\sum_{q=1}^{F_{l-1}} \mathbf{h}_l^{pq}(\mathbf{L}_n)\mathbf{x}_{l-1}^q \right) - \mathbf{P}_n \sigma\left(\sum_{q=1}^{F_{l-1}} \mathbf{h}_l^{pq}(\mathcal{L}) f_{l-1}^q \right) \right\|\end{aligned} \tag{28}$$

with $\mathbf{x}_0 = \mathbf{P}_n f$ as the input of the first layer. With a normalized Lipschitz nonlinearity, we have

$$\|\mathbf{x}_l^p - \mathbf{P}_n f_l^p\| \leq \left\| \sum_{q=1}^{F_{l-1}} \mathbf{h}_l^{pq}(\mathbf{L}_n)\mathbf{x}_{l-1}^q - \mathbf{P}_n \sum_{q=1}^{F_{l-1}} \mathbf{h}_l^{pq}(\mathcal{L}) f_{l-1}^q \right\| \tag{29}$$

$$\leq \sum_{q=1}^{F_{l-1}} \left\| \mathbf{h}_l^{pq}(\mathbf{L}_n)\mathbf{x}_{l-1}^q - \mathbf{P}_n \mathbf{h}_l^{pq}(\mathcal{L}) f_{l-1}^q \right\| \tag{30}$$

The difference can be further decomposed as

$$\begin{aligned}&\|\mathbf{h}_l^{pq}(\mathbf{L}_n)\mathbf{x}_{l-1}^q - \mathbf{P}_n \mathbf{h}_l^{pq}(\mathcal{L}) f_{l-1}^q\| \\ &\quad\leq \|\mathbf{h}_l^{pq}(\mathbf{L}_n)\mathbf{x}_{l-1}^q - \mathbf{h}_l^{pq}(\mathbf{L}_n)\mathbf{P}_n f_{l-1}^q \\ &\qquad + \mathbf{h}_l^{pq}(\mathbf{L}_n)\mathbf{P}_n f_{l-1}^q - \mathbf{P}_n \mathbf{h}_l^{pq}(\mathcal{L}) f_{l-1}^q\|\end{aligned} \tag{31}$$

$$\begin{aligned}&\leq \left\| \mathbf{h}_l^{pq}(\mathbf{L}_n)\mathbf{x}_{l-1}^q - \mathbf{h}_l^{pq}(\mathbf{L}_n)\mathbf{P}_n f_{l-1}^q \right\| \\ &\qquad + \left\| \mathbf{h}_l^{pq}(\mathbf{L}_n)\mathbf{P}_n f_{l-1}^q - \mathbf{P}_n \mathbf{h}_l^{pq}(\mathcal{L}) f_{l-1}^q \right\|\end{aligned} \tag{32}$$

The first term can be bounded as $\|\mathbf{x}_{l-1}^q - \mathbf{P}_n f_{l-1}^q\|$ with the initial condition $\|\mathbf{x}_0 - \mathbf{P}_n f_0\| = 0$. The second term can be denoted as D_{l-1}^n. With the iteration employed, we can have

$$\|\boldsymbol{\Phi}(\mathbf{H}, \mathbf{L}_n, \mathbf{P}_n f) - \mathbf{P}_n \boldsymbol{\Phi}(\mathbf{H}, \mathcal{L}, f)\| \leq \sum_{l=0}^{L} \prod_{l'=l}^{L} F_{l'} D_l^n.$$

Therefore, we can focus on the difference term D_l^n, we omit the feature and layer index to work on a general form, which leads to

$$\begin{aligned}&\|\mathbf{h}(\mathbf{L}_n)\mathbf{P}_n f - \mathbf{P}_n \mathbf{h}(\mathcal{L}) f\| \\ &\leq \left\| \sum_{i=1}^{n} h(\lambda_i^n)\langle \mathbf{P}_n f, \boldsymbol{\phi}_i^n \rangle \boldsymbol{\phi}_i^n - \sum_{i=1}^{\infty} h(\lambda_i)\langle f, \boldsymbol{\phi}_i \rangle \mathbf{P}_n \boldsymbol{\phi}_i \rangle \right\|\end{aligned} \tag{33}$$

$$\begin{aligned}&\leq \sum_{i=1}^{n} |h(\lambda_i^n)| \|\langle \mathbf{P}_n f, \boldsymbol{\phi}_i^n \rangle \boldsymbol{\phi}_i^n - \langle f, \boldsymbol{\phi}_i \rangle \mathbf{P}_n \boldsymbol{\phi}_i\| \\ &\quad + \|P_n f\| \sum_{i=1}^{n} |h(\lambda_i) - h(\lambda_i^n)| + \left\| \sum_{i=n+1}^{\infty} h(\lambda_i)\langle f, \boldsymbol{\phi}_i \rangle \mathbf{P}_n \boldsymbol{\phi}_i \right\|\end{aligned} \tag{34}$$

The first term in (34) can be bounded combined with the boundedness of frequency response as

$$\sum_{i=1}^{n} \langle \mathbf{P}_n f, \boldsymbol{\phi}_i^n \rangle \|\boldsymbol{\phi}_i^n - \mathbf{P}_n \boldsymbol{\phi}_i\| + \|\langle \mathbf{P}_n f, \boldsymbol{\phi}_i^n \rangle - \langle f, \boldsymbol{\phi}_i \rangle\| \|\boldsymbol{\phi}_i\| \tag{35}$$

With the results in Theorem 1 and $\lim_{n\to\infty} \mathbf{P}_n f = f$, the first term goes to 0 as n goes to infinity. The second term can be bounded by the Lipschitz continuity of the frequency response function, which leads to

$$\|P_n f\| \sum_{i=1}^{n} |h(\lambda_i) - h(\lambda_i^n)| \leq C \|P_n f\| \sum_{i=1}^{n} |\lambda_i - \lambda_i^n|, \tag{36}$$

as $n \to \infty$, $\lambda_i^n \to \lambda_i$. The last term also goes to 0 as $n \to \infty$. With $D_l^n \to 0$, this concludes the proof.