GRAPH CONVOLUTIONAL NETWORKS FROM THE PERSPECTIVE OF SHEAVES AND THE NEURAL TANGENT KERNEL

Thomas Gebhart
Department of Computer Science
University of Minnesota
Minneapolis, MN 55455, USA
gebhart@umn.edu

ABSTRACT

Graph convolutional networks are a popular class of deep neural network algorithms which have shown success in a number of relational learning tasks. Despite their success, graph convolutional networks exhibit a number of peculiar features, including a bias towards learning oversmoothed and homophilic functions, which are not easily diagnosed due to the complex nature of these algorithms. We propose to bridge this gap in understanding by studying the neural tangent kernel of sheaf convolutional networks–a topological generalization of graph convolutional networks. To this end, we derive a parameterization of the neural tangent kernel for sheaf convolutional networks which separates the function into two parts: one driven by a forward diffusion process determined by the graph, and the other determined by the composite effect of nodes' activations on the output layer. This geometrically-focused derivation produces a number of immediate insights which we discuss in detail.

1 INTRODUCTION

Graph neural networks (GNNs) are a class of deep learning architectures which aim to learn functions over relationally-structured data. GNNs come in a variety of forms Xu et al. (2018); Kipf & Welling (2016); Veličković (2022) which generally compute, within each layer, nonlinear functions of message-passing operations that update signals at each node in the input graph according to some notion of locality. The addition of deeper layers to these networks further propagates messages outwards from each node's local neighborhood, turning local message-passing operations into global transformations. The general and flexible computational structure provided by GNNs has led to their achievement of state-of-the-art performance on tasks spanning a variety of application domains from social science to drug design Chen et al. (2018); Liao et al. (2018).

Despite these successes, GNNs face a number of practical and theoretical shortcomings. For example, deep GNNs are known to over-smooth input data leading to the learning of rather generic function on the input graph signals which results in poor performance Chen et al. (2020). In addition, many types of GNNs are known to make strong assumptions about input graphs being homophilic–that connected nodes will be more similar to each other and will share more properties relative to other nodes in the network Zhu et al. (2020).

While architectural fixes for these types of problems have been proposed Zhu et al. (2020); Bodnar et al. (2022), the layer-wise nature of GNNs, in combination with the representational flexibility of graph-structured data, leads to substantial complexity and makes an intuitive understanding of the function of these algorithms elusive. Given these limitations, significant prior work has been devoted to these analytical tasks, resulting in a variety of approaches for characterizing the learning dynamics, bounding generalization performance, and formalizing the representational capacity of GNNs Du et al. (2019); Xu et al. (2018; 2020); Bodnar et al. (2022).

One approach for approximating both a formal and intuitive grasp on the behavior of neural networks in general is by analyzing their asymptotic behavior under gradient descent through the lens of the

neural tangent kernel of the network Jacot et al. (2018); Arora et al. (2019). In particular, Du et al. (2019) showed that the neural tangent kernel of graph convolutional networks (GCNs), a large and popular class of GNN architectures, can be described by a recursive relationship among the feature covariance of connected nodes. The authors were then able to use this interpretation to provide bounds on the asymptotic behavior of infinitely-wide GCNs and the function classes learnable by their studied architectures.

In this paper, we offer an extension to the study of GCNs through the their tangent kernels by developing a geometrically-oriented reformulation of their neural tangent kernel. We start by deriving the tangent kernel of a more general graph-convolutional architecture known as a sheaf convolutional network Hansen & Gebhart (2020). From the vantage point of this more general architecture, we gain the necessary perspective for reasoning about the shortcomings of GCNs through a kernel operator associated to sheaf convolutional networks. As we will observe, the resulting geometric reformulation is intuitive, providing insight into not only the functional behavior of GCNs, but also their idiosyncratic deficiencies. This reformulation emphasizes the susceptibility of deep GCNs to oversmoothing and provides a framework through which to make spectral arguments about their performance under particular distributions of structure over the input graphs. We also observe a relationship between the graph neural tangent kernel, and GCNs by extension, to more traditional diffusion-based graph kernels, charting a course for future work in designing relational deep learning architectures whose tangent kernels approximate more exotic graph kernels.

1.1 Notation

We denote vectors and matrices in bold script, respectively $\boldsymbol{x}$ and $\boldsymbol{X}$. Given a matrix $\boldsymbol{X}$, the submatrix corresponding to the rows in set R and columns in set C is denoted $\boldsymbol{X}[R, C]$, and $\boldsymbol{X}[\cdot, C]$ or $\boldsymbol{X}[R, \cdot]$ when $R = \emptyset$ or $C = \emptyset$, respectively. We use parentheses and subscripts $(\boldsymbol{X})_{r,c}$ to denote the choice of single elements in the r row and c column of $\boldsymbol{X}$. We will occasionally drop the parentheses when such notation is superfluous from context. The normal distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ is denoted $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

2 Sheaf Neural Networks

Sheaf neural networks were proposed by Hansen & Gebhart (2020) as a generalization of graph convolutional networks Kipf & Welling (2016) to cellular sheaf-structured data. Rooted in topology and homological algebra, cellular sheaves are a natural object through which to view signals over graph structures that are subject to particular constraints on the data between neighboring nodes. This additional structure provided by cellular sheaves affords sheaf neural networks the ability to distinguish classes of graphs that GCNs cannot while tempering structural issues like oversmoothing Hansen & Gebhart (2020); Bodnar et al. (2022). We begin with a brief overview of cellular sheaves and their spectral properties before introducing sheaf neural networks.

2.1 Cellular Sheaves

A *cellular sheaf* is an algebro-topological data structure which associates a graph's nodes and edges to data in another space. Formally, a cellular sheaf $\mathcal{F}$ on an undirected graph $G = (V, E)$ is specified by

- a vector space $\mathcal{F}(v)$ for each vertex $v \in V$
- a vector space $\mathcal{F}(e)$ for each edge $e \in E$, and
- a linear map $\boldsymbol{\mathcal{F}}_{v \trianglelefteq e} : \mathcal{F}(v) \to \mathcal{F}(e)$ for each incident vertex-edge pair $v \trianglelefteq e$ of G.

Sheaves impose consistency constraints on the data assigned to incident vertices across edges through the restriction maps $\boldsymbol{\mathcal{F}}_{v \trianglelefteq e}$. Specifically, given an edge e between vertices u and v, we say that a choice of data $\boldsymbol{x}_v \in \mathcal{F}(v)$, $\boldsymbol{x}_u \in \mathcal{F}(u)$ is *consistent* over e if $\boldsymbol{\mathcal{F}}_{v \trianglelefteq e}\boldsymbol{x}_v = \boldsymbol{\mathcal{F}}_{u \trianglelefteq e}\boldsymbol{x}_u$. The product space of data associated with all vertices of G is called the space of 0-cochains and is denoted $C^0(G; \mathcal{F})$. For our purposes, we may view $C^0(G; \mathcal{F})$ as a space of *signals* on the vertices of G, where the value of a signal at a vertex v lives in the vector space $\mathcal{F}(v)$. Similarly, we

denote the space of signals associated with edges by $C^1(G;\mathcal{F})$. Each edge of G imposes a constraint on $C^0(G;\mathcal{F})$ by restricting the space associated with its two incident vertices. The subspace of $C^0(G;\mathcal{F})$ satisfying all these constraints is the space of *global sections* of $\mathcal{F}$, and is denoted $H^0(G;\mathcal{F})$. Data in $H^0(G'\mathcal{F})$ are the assignments which satisfy the constraints introduced by $\mathcal{F}$ on G.

The space of global sections $H^0(G;\mathcal{F})$ is the kernel of a linear map $\boldsymbol{\delta} : C^0(G;\mathcal{F}) \to C^1(G;\mathcal{F})$ called the *coboundary*, and, given an arbitrary choice of orientation on the edges of the graph, may be computed by

$$(\boldsymbol{\delta}\boldsymbol{x})_e = \boldsymbol{\mathcal{F}}_{v \trianglelefteq e}\boldsymbol{x}_v - \boldsymbol{\mathcal{F}}_{u \trianglelefteq e}\boldsymbol{x}_u$$

for each oriented edge $e = u \to v$. Therefore, if $\boldsymbol{x} \in \ker \boldsymbol{\delta}$, then $\boldsymbol{\mathcal{F}}_{v \trianglelefteq e}\boldsymbol{x}_v = \boldsymbol{\mathcal{F}}_{u \trianglelefteq e}\boldsymbol{x}_u$ for every edge $e = u \sim v$.

Sheaves have their own Laplacian operator Hansen & Ghrist (2019) which simplifies to the graph Laplacian when the constraints imposed by the restriction maps are lifted such that $\boldsymbol{\mathcal{F}}_{v \trianglelefteq e} = \boldsymbol{I}$ for all $v \in V, e \in E$. The construction of the sheaf Laplacian mirrors the approach for the graph Laplacian as the matrix product of incidence matrices. Given a coboundary operator, the *sheaf Laplacian* is given by $\boldsymbol{L}_{\mathcal{F}} = \boldsymbol{\delta}^T\boldsymbol{\delta}$, which is a positive semidefinite linear operator on $C^0(G;\mathcal{F})$ with kernel $H^0(G;\mathcal{F})$.

Rather than simply recording connections between nodes, cellular sheaves specify relationships between data associated with those nodes. Standard graph-theoretic constructions like Laplacians and diffusion operators implicitly work with the *constant sheaf* on a graph: the sheaf $\underline{\mathbb{R}}$ with all stalks $\mathbb{R}$ and all restriction maps the identity. This is a simple relationship between nodes which can be greatly generalized in the sheaf setting. For instance, a sheaf can easily represent a signed graph by changing the sign of one restriction map of the constant sheaf for each negatively signed edge. More general relationships between nodes can be expressed, especially as stalks increase in dimension, resulting in such operators as connection Laplacians Singer & Wu (2012) and matrix-weighted Laplacians Tuna (2016). These generalizations have just begun to be explored in the context of graph learning Hansen & Gebhart (2020); Bodnar et al. (2022).

2.2 Diffusion and Sheaf Convolutional Networks

Graph convolutional networks Kipf & Welling (2016) exploit local graph diffusion operators to define a neighborhood convolution operation over nodes, acting as a generalization of convolutional neural networks over an irregular domain defined by a graph. These graph diffusion operations are typically implemented through a transformation of the graph's adjacency matrix $\boldsymbol{A}$ or, for GCNs in particular, the normalized Laplacian matrix $\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$ where $\boldsymbol{D}$ is the degree matrix of G.

Sheaf convolutional networks employ an analogous diffusion operation to define convolution-like operations of signals on $\mathcal{F}$. The sheaf Laplacian $\boldsymbol{L}_{\mathcal{F}} = \boldsymbol{\delta}^T\boldsymbol{\delta}$ encodes this diffusion operation in sheaf convolutional networks and, like the graph Laplacian, fulfills a number of desirable properties. The zero eigenspace of $\boldsymbol{L}_{\mathcal{F}}$ corresponds to the sections of $\mathcal{F}$, meaning we may interpret this operator's spectral structure as providing information on the signals which represent consistent assignments of data given the sheaf constraints. For an appropriately chosen α, the operator $\boldsymbol{Q}^{\alpha}_{\mathcal{F}} = \boldsymbol{I} - \alpha\boldsymbol{L}_{\mathcal{F}}$ will have 2-norm 1 and has $H^0(G;\mathcal{F})$ as the eigenspace corresponding to the eigenvalue 1. There is also a normalized form of the sheaf Laplacian, $\tilde{\boldsymbol{L}}_{\mathcal{F}} = \boldsymbol{D}^{-1/2}\boldsymbol{L}_{\mathcal{F}}\boldsymbol{D}^{-1/2}$, where $\boldsymbol{D}$ is the block diagonal of $\boldsymbol{L}_{\mathcal{F}}$. This normalization effectively reparameterizes the stalks of $\mathcal{F}$ so that the eigenvalues are bounded below by 0 and above by 2. Therefore, we can also construct a stable diffusion operator $\tilde{\boldsymbol{Q}}_{\mathcal{F}} = \boldsymbol{I} - \tilde{\boldsymbol{L}}_{\mathcal{F}}$. Diffusion operators depending on larger neighborhoods may be constructed from powers of these operators. For any l, $\tilde{\boldsymbol{Q}}^l_{\mathcal{F}}$ and $(\boldsymbol{Q}^{\alpha}_{\mathcal{F}})^l$ are l-step sheaf diffusion operators.

With this information, we can now define a sheaf convolutional layer. Assume our input data comes in the form of signals $\boldsymbol{X}$ over a sheaf $\mathcal{F}$. As this is sheaf-structured data, these signals may be multi-dimensional such that each node $v \in V$ has a variable-dimensional stalk $\mathcal{F}(v)$. For simplicity, we will assume each stalk is k-dimensional. In addition, we assume each node signal has d channels, forming a d-dimensional vector in $\mathbb{R}^d$. When $k = 1$, $\boldsymbol{X}$ is a $(N_V \times d)$ matrix and assumes the form of the more familiar feature matrix in standard graph learning. When $k > 1$, $\boldsymbol{X}$ has shape $N_V k \times d$ and contains N_v blocks of k rows which correspond to the k-dimensional stalk assignments of features over each node with the d channels of each of these k-dimensional features comprising the columns.

Definition 2.1. Let $\boldsymbol{\mathcal{D}}$ be some diffusion operator for the sheaf $\mathcal{F}$ such as $\boldsymbol{\mathcal{D}} = \boldsymbol{I} - \tilde{\boldsymbol{L}}_{\mathcal{F}}$. Given a choice of nonlinearity σ, a *sheaf convolutional layer* for sheaf signals $\boldsymbol{X}_l$ in layer l is defined as

$$\boldsymbol{X}_{l+1} = \sigma(\boldsymbol{\mathcal{D}}(\boldsymbol{I} \otimes \boldsymbol{B}_l)\boldsymbol{X}_l\boldsymbol{W}_l$$

where $\boldsymbol{W}_l$ is a $(d_{l+1} \times d_l)$ weight matrix, $\boldsymbol{B}_l$ a $(k \times k)$ weight matrix, and $(\boldsymbol{I} \otimes \boldsymbol{B})$ the Kronecker product of an $(N_V \times N_V)$ identity matrix with $\boldsymbol{B}$.

The right multiplication of $\boldsymbol{W}_l$ with $\boldsymbol{X}_l$ enacts a linear transformation of the channels in an equivalent manner to traditional GCNs. The left multiplication of $(\boldsymbol{I} \otimes \boldsymbol{B}_l)$ allows for a linear transformation of each channel's feature vector by multiplying each row block of $\boldsymbol{X}_l$ by $\boldsymbol{B}_l$. When $\mathcal{F}$ is the constant sheaf, $k = 1$, and $\boldsymbol{B}_l$ the identity matrix, the sheaf convolutional layer corresponds to a traditional graph-convolutional layer. We may compose these sheaf convolutional layers to create a sheaf convolutional network with L layers via

$$f(\boldsymbol{W}, \boldsymbol{B}, G)(\boldsymbol{X}) = \sigma(\boldsymbol{\mathcal{D}}(\boldsymbol{I} \otimes \boldsymbol{B}_L) \cdots \sigma(\boldsymbol{\mathcal{D}}(\boldsymbol{I} \otimes \boldsymbol{B}_2)\sigma(\boldsymbol{\mathcal{D}}(\boldsymbol{I} \otimes \boldsymbol{B}_1)\boldsymbol{X}\boldsymbol{W}_1)\boldsymbol{W}_2) \cdots \boldsymbol{W}_L). \quad (1)$$

Note this architecture is composed only of sheaf convolutional layers. In graph classification tasks, it is common to add a "readout" operation ρ at the final layer which combines the latent node features to create an output classification or regression prediction for the task. A simple example of such a readout function is sum-aggregation $\rho(\boldsymbol{X}_L) = \sum_{i=1}^{N_V}(\boldsymbol{X}_L\boldsymbol{1})_v$. In node classification tasks, the readout function may take the form of a sigmoid operation which maps the hidden node representations themselves to output class or regression values instead of aggregating over nodes Xu et al. (2018).

3 Neural Tangent Kernels

We now turn our attention to graph neural tangent kernels and their extension to sheaves. After introducing the graph neural tangent kernel, we will derive a geometrically-focused parameterization of the neural tangent kernel for the sheaf convolutional network architecture given in the previous section.

The neural tangent kernel Jacot et al. (2018) for fully-connected and convolutional networks describes the behavior and asymptotic performance of these networks under the assumption that they are trained by gradient descent with an infinitesimally small learning rate, are initialized randomly, and have layers of infinite width. Under these assumptions, one can replace these infinitely-wide networks with a deterministic kernel machine whose kernel is given by

$$\boldsymbol{\Theta}(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}_{\boldsymbol{W} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})}\left[\left\langle \frac{\partial f(\boldsymbol{W}, \boldsymbol{x})}{\partial \boldsymbol{W}}, \frac{\partial f(\boldsymbol{W}, \boldsymbol{x}')}{\partial \boldsymbol{W}} \right\rangle\right]$$

where $\boldsymbol{x}, \boldsymbol{x}'$ are two inputs and f is a feedforward neural network with parameters $\boldsymbol{W}$. Du et al. (2019) showed that this notion of the neural tangent kernel extends to graph neural network architectures, and the authors provide a recursive formulation termed the graph neural tangent kernel. We will use this as our starting point for describing a geometrically-focused parameterization of the neural tangent kernel for sheaf neural networks and, by extension, graph convolutional networks.

3.1 A Sheaf Neural Tangent Kernel

Given a set of n input sheaves $\{\mathcal{F}_i\}_{i=1}^n$, the supervised sheaf learning task looks to learn a sheaf convolutional network f parameterized by weights $\boldsymbol{W}$ and $\boldsymbol{B}$ as structured in Equation 1 with an additional readout function ρ. We would like to study the neural tangent kernel corresponding to this network, given by

$$\boldsymbol{\Theta}(\mathcal{F}, \mathcal{F}') = \mathbb{E}_{\boldsymbol{W} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})}\left[\left\langle \frac{\partial f(\boldsymbol{W}, \mathcal{F})}{\partial \boldsymbol{W}}, \frac{\partial f(\boldsymbol{W}, \mathcal{F}')}{\partial \boldsymbol{W}} \right\rangle\right]$$

for two input cellular sheaves $\mathcal{F}, \mathcal{F}'$ based on graphs $G = (V, E)$ and $G' = (V', E')$ with the same number of nodes $|V| = |V'| = N_V$ but potentially differing number of edges. In addition, we assume that $\mathcal{F}$ and $\mathcal{F}'$ have the same stalk dimensionality k, resulting in sheaf signals $\boldsymbol{X}, \boldsymbol{X}'$ of size $(N_v k \times d)$. Sheaves $\mathcal{F}$ and $\mathcal{F}'$ give rise to potentially distinct diffusion operators $\boldsymbol{\mathcal{D}}, \boldsymbol{\mathcal{D}}'$. Finally, we assume $\boldsymbol{B}$ and $\boldsymbol{B}'$ are fixed across layers of f and simplify notation by integrating these

constant matrices into the diffusion operators $\boldsymbol{\mathcal{D}}$ and $\boldsymbol{\mathcal{D}}'$, respectively. Taking the partial derivative with respect to the parameters in layer l, we see

$$\frac{\partial f(\boldsymbol{W}, \mathcal{F})}{\partial \boldsymbol{W}_l} = \left(\boldsymbol{\mathcal{D}}\sigma(\boldsymbol{f}_l)\right)^\top \boldsymbol{P}_l$$

where $\boldsymbol{P}_l = \boldsymbol{\mathcal{D}}^\top \boldsymbol{P}_{l+1} \boldsymbol{W}_{l+1}^\top \odot \dot{\sigma}(\boldsymbol{f}_l)$ captures the gradient propagating backwards from deeper network layers. Note the slight abuse of notation by viewing the bold-cased $\boldsymbol{f}_l$ as the vector-valued output of the l-layer sheaf convolutional network f_l. Denoting the inner product $\boldsymbol{\theta}(f)$, we have

$$\begin{aligned}
\boldsymbol{\theta}(f) &= \left\langle \frac{\partial f(\boldsymbol{W}, \mathcal{F})}{\partial \boldsymbol{W}}, \frac{\partial f(\boldsymbol{W}, \mathcal{F}')}{\partial \boldsymbol{W}} \right\rangle \\
&= \left\langle \left(\boldsymbol{\mathcal{D}}\sigma(\boldsymbol{f}_l)\right)^\top \boldsymbol{P}_l, \left(\boldsymbol{\mathcal{D}}'\sigma(\boldsymbol{f}'_l)\right)^\top \boldsymbol{P}'_l \right\rangle \\
&= \left\langle \left(\boldsymbol{\mathcal{D}}\sigma(\boldsymbol{f}_l)\right)^\top, \left(\boldsymbol{\mathcal{D}}'\sigma(\boldsymbol{f}'_l)\right)^\top \right\rangle \odot \langle \boldsymbol{P}_l, \boldsymbol{P}'_l \rangle \\
&= \boldsymbol{\Sigma}_l \odot \langle \boldsymbol{P}_l, \boldsymbol{P}'_l \rangle.
\end{aligned} \tag{2}$$

Focusing on the left side of the Hadamard product in Equation 2, we see a structure akin to the covariance of diffusion operations applied to the node representations at each layer. In the input layer, $\boldsymbol{\Sigma}_0 = \boldsymbol{X}\boldsymbol{X'}^\top$ is precisely the channel covariance of the input signals, and similarly, $\boldsymbol{\Sigma}_1 = \boldsymbol{\mathcal{D}}\boldsymbol{X}\boldsymbol{X}'^\top\boldsymbol{\mathcal{D}}'^\top$. For layers $l > 1$, more care is required to write $\boldsymbol{\Sigma}_l$ in closed form due to the introduction of weights and nonlinearities in the message passing updates. To arrive at a representation for $\boldsymbol{\Sigma}_l$, we will take expectations of layer-wise activations under the assumption that the network layers of f are infinitely wide ($d_l \to \infty$) and weights are initialized according to a standard normal distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. These assumptions lead to the following proposition.

Proposition 3.1. *Given an infinitely-wide sheaf convolutional network with parameters initialized according to a standard normal distribution, element-wise activation function σ, and sheaf signals $\boldsymbol{X}, \boldsymbol{X}'$ over sheaves with diffusion operators $\boldsymbol{\mathcal{D}}, \boldsymbol{\mathcal{D}}'$, the covariance of the diffusion operation gradients at layer l is defined recursively as*

$$\boldsymbol{\Sigma}_l = \boldsymbol{\mathcal{D}}\boldsymbol{H}_{l-1}\boldsymbol{\mathcal{D}}'^\top$$

where

$$\boldsymbol{H}_{l-1} = \mathop{\mathbb{E}}_{\boldsymbol{f}_{l-1}, \boldsymbol{f}'_{l-1} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_{l-1})} \left[\sigma(\boldsymbol{f}_{l-1})\sigma(\boldsymbol{f}'_{l-1})^\top\right].$$

The proof of this proposition follows from the recursive definition given in Du et al. (2019), but a full derivation may be found in the appendix of the arXiv version of this paper[1]. Combining Proposition 3.1 with the inner product between the deeper layers' backpropagated gradients according to Equation 2 results in a geometrically-oriented parameterization of the sheaf neural tangent kernel.

Proposition 3.2. *For an L-layer sheaf convolutional network structured as in Equation 1 with fixed $\boldsymbol{B}$ at each layer, its corresponding neural tangent kernel between two sheaves $\mathcal{F}$ and $\mathcal{F}'$ may be written*

$$\begin{aligned}
\boldsymbol{\Theta}(\mathcal{F}, \mathcal{F}') &= \sum_{l=1}^{L+1} \left(\boldsymbol{\Sigma}_l \odot (\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{D}}'^\top)^{\odot(L+1-l)}\right) \odot \left(\bigodot_{i=l}^{L+1-l} \dot{\boldsymbol{H}}_i\right) \\
&= \sum_{l=1}^{L+1} \boldsymbol{\Delta}_l \odot \boldsymbol{\Pi}_l
\end{aligned} \tag{3}$$

where $\dot{\boldsymbol{H}}_l = \mathbb{E}_{\boldsymbol{f}_l, \boldsymbol{f}'_l \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_l)}[\dot{\sigma}(\boldsymbol{f}_l)\dot{\sigma}(\boldsymbol{f}'_l)^\top]$, $\boldsymbol{\Delta}_l = \boldsymbol{\Sigma}_l \odot (\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{D}}'^\top)^{\odot(L+1-l)}$, and $\boldsymbol{\Pi}_l = \bigodot_{i=l}^{L+1-l} \dot{\boldsymbol{H}}_i$. The proof of this proposition may again be found in the appendix of the arXiv version of this paper. This proof aligns with previous work on graph convolutional networks when the sheaf stalk dimensionality $k = 1$ Sabanayagam et al. (2021). Note that in the node classification setting $\boldsymbol{\mathcal{D}} = \boldsymbol{\mathcal{D}}'$.

Equation 3 clearly delineates the structure of $\boldsymbol{\Theta}$ as the sum of element-wise products of diffusion-related effects $\boldsymbol{\Delta}_l$ with parameter path activation effects $\boldsymbol{\Pi}_l$. To further clarify this relationship,

[1] Available at https://arxiv.org/abs/2208.09309.

consider the case when the activation function σ is the identity. Under this scenario, each $\dot{\boldsymbol{H}}_i$ is $\mathbf{1}$, and the tangent kernel becomes

$$\begin{aligned}\bar{\boldsymbol{\Theta}}(\mathcal{F}, \mathcal{F}') &= \sum_{l=1}^{L+1} \boldsymbol{\mathcal{D}}^l \boldsymbol{X} \boldsymbol{X}'^{\top} (\boldsymbol{\mathcal{D}}'^{\top})^l \odot (\boldsymbol{\mathcal{D}} \boldsymbol{\mathcal{D}}'^{\top})^{\odot(L+1-l)} \\ &= \sum_{l=1}^{L+1} \bar{\boldsymbol{\Delta}}_l.\end{aligned} \tag{4}$$

The neural tangent kernel for such a linear sheaf convolutional network lacks the dependence on the activation paths and takes the form of a weighted sum over products of the l-step diffusion of signals on sheaves $\mathcal{F}$ and $\mathcal{F}'$.

4 Discussion

The derivation of the sheaf neural tangent kernel given in Equation 3 produces a number of intuitive insights which assist in the interpretation of both the function and limitations of sheaf convolutional networks and, by extension, graph convolutional networks.

4.1 Relationship to Graph Convolutional Networks

When the input space is structured such that $\mathcal{F}, \mathcal{F}'$ are both constant sheaves with 1-dimensional stalks ($k = 1$), Equation 3 is equivalent to the graph neural tangent kernel between graphs G and G' with diffusion matrices given by, for example, adjacency matrices $\boldsymbol{A}, \boldsymbol{A}'$ or their transformations as respective graph Laplacians $\boldsymbol{\mathcal{D}} = (\boldsymbol{I} - \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{A} \boldsymbol{D}^{-\frac{1}{2}})$, $\boldsymbol{\mathcal{D}}' = (\boldsymbol{I} - \boldsymbol{D}'^{-\frac{1}{2}} \boldsymbol{A}' \boldsymbol{D}'^{-\frac{1}{2}})$ Sabanayagam et al. (2021). In other words, graph convolutional networks operate on sheaves with trivial structure. This also implies $\bar{\boldsymbol{\Theta}}$ for trivial sheaf structures is closely related to the neural tangent kernel of the simplified graph convolutional network architecture described in Wu et al. (2019). These *simple graph convolutional networks* are structured as

$$f^{\text{smp}}(\boldsymbol{W}, G, L)(\boldsymbol{X}) = \rho(\boldsymbol{\mathcal{D}}^L \boldsymbol{X} \boldsymbol{W}).$$

Therefore, the corresponding neural tangent kernel is given by

$$\bar{\boldsymbol{\Theta}}_L^{\text{smp}} = \boldsymbol{\mathcal{D}}^L \boldsymbol{X} \boldsymbol{X}'^{\top} (\boldsymbol{\mathcal{D}}'^{\top})^L \odot (\boldsymbol{\mathcal{D}} \boldsymbol{\mathcal{D}}'^{\top}).$$

The neural tangent kernel for linear graph convolutional networks is composed of weighted sums of $\bar{\boldsymbol{\Theta}}_l^{\text{smp}}$ for each layer l. This close approximation is reflected by the empirical results of Wu et al. (2019) which show that simple graph convolutional networks can achieve performance in line with those of more complicated graph neural network architectures on particular tasks. These results provide some confidence that even our linearized sheaf neural tangent kernel may still provide insight into the behavior of the more complex graph convolutional network architectures used in practice.

4.2 Oversmoothing

As observed by Bodnar et al. (2022), the trivial structure imposed by GCNs is intimately related to the tendency of these architectures to bias towards learning over-smoothed, homophilic representations of the input graph signals. Our neural tangent kernel derivation provides further confirmation of the presence of these biases in GCNs.

Equation 4 shows that, for a network of depth L, the kernel value $\bar{\boldsymbol{\Theta}}(G, G')$ between graph signals $\boldsymbol{X}, \boldsymbol{X}'$ on graphs G and G' will contain terms consisting of signals whose feature values have been diffused across $\{1, 2, \ldots, L\}$ steps along the graph. Although the deeper diffusion terms are exponentially down-weighted according to the element-wise power of $\boldsymbol{\mathcal{D}} \boldsymbol{\mathcal{D}}'^{\top}$, their oversmoothing effects on deep graph convolutional networks are worrisome due to the fact that $\lim_{L \to \infty} \boldsymbol{\mathcal{D}}^L$ will approach an orthogonal projection onto the harmonic space $H^0(G, \underline{\mathcal{F}})$. This subspace consists of constant functions on G which lack discriminative power over the nodes of G. As a result, $\boldsymbol{\Theta}(G, G')$ for a sufficiently deep graph convolutional network will approximate the inner product of the projection of signals $\boldsymbol{X}$ and $\boldsymbol{X}'$ onto the kernel of $\boldsymbol{\mathcal{D}}$ and $\boldsymbol{\mathcal{D}}'$. For exceptionally deep networks,

we can view $\bar{\boldsymbol{\Theta}}(G, G')$ as returning the similarity between the limiting distributions of random walks on G and G'. Spectral bounds for such random walk processes are well-studied Chung & Graham (1997); Lovász (1993), and the incorporation of such methods in analyzing the discriminatory power of graph neural tangent kernels may prove insightful for future work.

We also view the homophilic bias imparted by the trivial diffusion structure of GCNs through this spectral lens. To see this, assume the setup of a 2-class node classification task such that the class assignments $\boldsymbol{y} \in \{-1, 1\}$ partition of the nodes of G. We can bound the mixing time $M(G)$ of G by

$$M(G) \sim O\left(\frac{\log \min_v(\boldsymbol{\pi}(G))_v^{-1}}{h(G)^2}\right) \tag{5}$$

where $\boldsymbol{\pi}(G)$ the limiting distribution of G and $h(G)$ is the Cheeger constant:

$$h(G) = \min_S \left\{\frac{|\partial S|}{\min\{|S|, |\overline{S}|\}}\right\}$$

where $S \subset G$ and ∂S is the edge boundary of G composed of the edges which connect nodes in S and $\overline{S}$. The bound in Equation 5 is maximized for homophilic graphs which are composed of a small number of densely-connected clusters with weak between-cluster connectivity. Such graphs have small $h(G)$ and will mix more slowly over a fixed number of diffusion steps, leading to increased separation in the tangent kernel between nodes of different classes for L fixed (assuming the class assignments $\boldsymbol{y}$ respect the homophilic structure).

The bound on $M(G)$ is minimized when G is a complete bipartite graph, causing the diffusion process executed by a graph convolutional network to approach harmonicity at an even faster rate and decreasing the separability of the neural tangent kernel. Worse, when G is a connected bipartite graph and the distribution of class labels is opposite across two equal node partitions of G, the diffusion process reaches a steady state immediately, sending the node representations to the kernel of $\boldsymbol{\mathcal{D}}$ and trivializing $\bar{\boldsymbol{\Theta}}$. By contrast, consider the same same assumptions on G, but with $\boldsymbol{\mathcal{D}}$ composed of restriction maps with opposing signs on each incident edge $\mathcal{F}_{v \trianglelefteq e} = -\mathcal{F}_{u \trianglelefteq e}$. Signals diffusing over this sheaf oscillate across nodes in each bipartition instead of being immediately sent to the kernel of $\boldsymbol{\mathcal{D}}$.

These observations, which align with the results on the linear separability of sheaf diffusion discussed in Bodnar et al. (2022), reveal that the constraints imposed by sheaf structures are crucial to learning over particular graph structures when applying convolutional architectures. Through the addition of proper sheaf constraints, one can ameliorate the tendency of graph convolutional networks to mix too quickly over particular graph structures and consequently learning simplistic node representations. Unfortunately, the necessary constraints are typically unknown *a priori*. Although it may be possible to learn these constraints from raw graph signals themselves Bodnar et al. (2022), the question of which sheaf constraints are optimal for a given input graph structure remains an important open question.

4.3 Relationship to Diffusion Kernels

As noted in the previous section, the linear sheaf neural tangent kernel $\bar{\boldsymbol{\Theta}}$ is determined by the weighted sum of diffusion-like operations. The use of diffusion as a graph kernel is a well-studied topic in the traditional graph kernel literature Smola & Kondor (2003). Using this graph kernel language, the structure of $\bar{\boldsymbol{\Theta}}$ within each layer l may be described as the composition of two l-step random walk kernels $\boldsymbol{K} = (\alpha \boldsymbol{I} - \tilde{\boldsymbol{L}})$ Kondor & Lafferty (2002) and an inner product kernel. This relationship offers an interesting avenue for future work in determining the extent to which neural network architectures may be augmented such that their neural tangent kernels approximate compositions of more exotic graph kernels.

4.4 Influence of Parameter Connectivity

Our discussion thus far has focused on the diffusion terms $\boldsymbol{\Delta}_l$ of of the tangent kernel as given in Equation 3. However, the effects of parameter paths as encoded by $\boldsymbol{\Pi}_l$ on $\boldsymbol{\Theta}$ cannot be ignored, especially when $\sigma(x) = \max\{0, x\}$.

With ReLU activation functions, each $\mathbf{\Pi}_l$ acts as a mask on the diffusion kernel values in $\mathbf{\Delta}_l$, zeroing diffusion kernel values between nodes u and v for if their weighted activation becomes negative at any proceeding layer. In this way, $\mathbf{\Theta}$ is the element-wise product of two descriptions of connectivity between u and v: one coming from the similarity of an l-step diffusion of their signals along the graph, and the other the weighted connectedness of their features through $\boldsymbol{W}$.

Under the neural tangent kernel assumptions, $\boldsymbol{W}$ is normally distributed and does not change during training. Abusing this model slightly, we can hypothesize that finite-width networks will adjust $\boldsymbol{W}$ to weight distinctly features resulting from diffusion operations at different layers. In other words, ReLU activations allow the network to control which steps of the random walk it attends to given an input signal on the graph, thereby learning the important degrees of locality for a task. Although in practice it appears that neural tangent kernels can approximate the behavior of finite-width networks Lee et al. (2020), more work is required to show how this approximation behaves as layer width decreases and how this movement into the feature regime affects our understanding of graph and sheaf neural tangent kernels as being driven by diffusion processes on the underlying network.

REFERENCES

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *Advances in Neural Information Processing Systems*, 32, 2019.

Cristian Bodnar, Francesco Di Giovanni, Benjamin Paul Chamberlain, Pietro Liò, and Michael M Bronstein. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in gnns. *arXiv preprint arXiv:2202.04579*, 2022.

Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: Fast learning with graph convolutional networks via importance sampling. In *International Conference on Learning Representations*, 2018.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, pp. 1725–1735. PMLR, 2020.

Fan RK Chung and Fan Chung Graham. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.

Simon S Du, Kangcheng Hou, Russ R Salakhutdinov, Barnabas Poczos, Ruosong Wang, and Keyulu Xu. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. *Advances in neural information processing systems*, 32, 2019.

Jakob Hansen and Thomas Gebhart. Sheaf neural networks. In *NeurIPS 2020 Workshop on Topological Data Analysis and Beyond*, 2020.

Jakob Hansen and Robert Ghrist. Toward a spectral theory of cellular sheaves. *Journal of Applied and Computational Topology*, 3(4):315–358, 2019.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Risi Imre Kondor and John Lafferty. Diffusion kernels on graphs and other discrete structures. In *Proceedings of the 19th international conference on machine learning*, volume 2002, pp. 315–322, 2002.

Jaehoon Lee, Samuel Schoenholz, Jeffrey Pennington, Ben Adlam, Lechao Xiao, Roman Novak, and Jascha Sohl-Dickstein. Finite versus infinite neural networks: an empirical study. *Advances in Neural Information Processing Systems*, 33:15156–15172, 2020.

Renjie Liao, Zhizhen Zhao, Raquel Urtasun, and Richard Zemel. Lanczosnet: Multi-scale deep graph convolutional networks. In *International Conference on Learning Representations*, 2018.

László Lovász. Random walks on graphs. *Combinatorics, Paul erdos is eighty*, 2(1-46):4, 1993.

Mahalakshmi Sabanayagam, Pascal Esser, and Debarghya Ghoshdastidar. New insights into graph convolutional networks using neural tangent kernels. *arXiv preprint arXiv:2110.04060*, 2021.

Amit Singer and Hau-Tieng Wu. Vector Diffusion Maps and the Connection Laplacian. *Communications in Pure and Applied Mathematics*, 65(8), 2012.

Alexander J Smola and Risi Kondor. Kernels and regularization on graphs. In *Learning theory and kernel machines*, pp. 144–158. Springer, 2003.

S. Emre Tuna. Synchronization under matrix-weighted Laplacian. *Automatica*, 73:76–81, November 2016. ISSN 0005-1098.

Petar Veličković. Message passing all the way up. *arXiv preprint arXiv:2202.11097*, 2022.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pp. 6861–6871. PMLR, 2019.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Keyulu Xu, Mozhi Zhang, Jingling Li, Simon S Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. How neural networks extrapolate: From feedforward to graph neural networks. *arXiv preprint arXiv:2009.11848*, 2020.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*, 33:7793–7804, 2020.