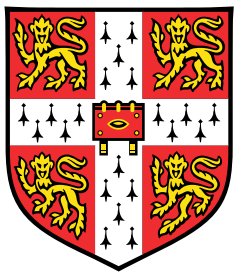
Attention-based Sheaf Neural Networks



Federico Barbero

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Master of Philosophy

King's College August 2022

Dedicated to my family.

DECLARATION

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 15,000 words (14,821 words total) including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

The software for the project originates from a Twitter Cortex repository with Apache 2.0 licensing, which is a permissive free software license, supplied by Cristian Bodnar. Significant modifications have been made to the codebase, including the implementation of Sheaf Attention Networks and Attentive Neural Sheaf Diffusion, which involves additional unit tests and evaluation pipelines which were required for the project.

The code may be found in the MLSALT machines in the directory /homes/fb548/mphil-thesis.

Federico Barbero
August 2022

ACKNOWLEDGEMENTS

This thesis is the culmination of a very intense, challenging, and rewarding year. I am very grateful to have been given this opportunity and to have experienced it alongside many wonderful people, making my year in Cambridge one I will hold dear to my heart.

I want to start by thanking Professor Pietro Liò. Your contagious passion, kindness, and humour made me feel immediately at home in this quaint medieval town. It is not surprising that you have managed to form such a brilliant group of exceptional students. One of these students is Cristian Bodnar, to whom I am particularly indebted. Thank you for all the time you've given up supervising me. You have been an incredible mentor throughout the year, and I wish you the greatest success. I hope we can keep up the cool topology stuff :-)

I'd also like to thank the Systems Security crew: Professor Lorenzo Cavallaro, Dr. Fabio Pierazzi, and Dr. Feargus Pendlebury for the beautiful time in San Francisco at IEEE S&P and Professor Michael Bronstein for the fantastic opportunity to go to Baltimore for ICML this year.

The challenging year was made significantly more bearable in large part because of my parents, Giovanna and Roberto, my brother Edoardo and my girlfriend Mei Lan. Thank you for your constant support, especially in the most demanding periods. A thank you to the many MLMI students I am now honoured to call my friends. You all will end up doing amazing things, and I am excited to hear all about it. Of course, thank you to the MLMI staff as well; you definitely know how to organise a challenging course.

To the reader: I hope that you will enjoy my thesis ;) I am off on a long-awaited break until something else comes up :-)

ABSTRACT

Graph Neural Networks (GNNs) offer a principled way of tackling machine learning tasks over graph structures. Despite their great success, they demonstrate some pathological issues. For example, their performance tends to suffer as more layers are added (oversmoothing), and they tend to perform worse in heterophilic graphs, in which nodes of different types tend to be connected (heterophily problem). Topological approaches involving cellular sheaves, giving rise to Sheaf Neural Networks (SNNs), have shown to be very promising in tackling these two issues. Building upon the recent success of SNNs and the wide adoption of attention-based architectures, we propose Attention-based Sheaf Neural Networks, a generalised attention mechanism that operates over cellular sheaves. We show that this type of construction generalizes popular attention-based GNN models to cellular sheaves and demonstrate that theoretically and empirically, these models help tackle the issues related to oversmoothing and heterophily. Our new models, Sheaf Attention Networks and Attentive Neural Sheaf Diffusion demonstrate great promise on a wide range of synthetic and real-world benchmarks. Overall, this work provides an exciting connection between attention mechanisms and algebraic topology, and hope it will help promote the further adoption and research of Sheaf Neural Networks.

TABLE OF CONTENTS

List of figures **viii**

List of tables **x**

1 Introduction **1**
1.1 Contributions . . . 3
1.2 Outline . . . 3

2 Background **5**
2.1 Why should we use geometry? . . . 5
2.2 Graph Neural Networks . . . 7
2.2.1 Graph Convolutional Networks . . . 8
2.2.2 Graph Attention Networks . . . 10
2.2.3 Graph Neural Diffusion . . . 12
2.2.4 Oversmoothing and Heterophily . . . 13
2.3 Cellular Sheaf Theory . . . 16
2.3.1 Sheaf Laplacians . . . 18
2.3.2 $O(d)$-bundles and Connection Laplacians . . . 20
2.4 Sheaf Diffusion . . . 22
2.4.1 Neural Sheaf Diffusion . . . 23

3 Attention Mechanisms over Sheaves **27**
3.1 Attentive Sheaf Diffusion . . . 28
3.2 Sheaf Attention Networks . . . 32
3.3 Attentive Neural Sheaf Diffusion . . . 35

3.4 Evaluation 36
3.4.1 Stalk Width 38
3.4.2 Oversmoothing 39
3.4.3 Heterophily 41
3.4.4 Molecular Applications 45
3.4.5 Limitations 47

4 Conclusion **51**
4.1 Future Work 52

References **53**

Appendix A Mathematical Material **57**
A.1 Graphs and Topology 57
A.2 Supplementary Proofs 58

Appendix B Experiments **60**
B.1 Datasets 60
B.1.1 Real-World 60
B.2 Synthetic 61
B.3 Hyper-parameters 63

LIST OF FIGURES

2.1 Accuracy on the cora dataset as more GCN layers are added. 14

2.2 Accuracy of a GCN on a synthetic benchmark with varying node homophily level. Low h means low homophily (high heterophily), high h means high homophily (low heterophily). 15

2.3 A diagrammatic representation of a sheaf $(G, \mathcal{F})$ for a single edge. The node *stalks* $\mathcal{F}(v)$ and $\mathcal{F}(u)$ and the edge stalk $\mathcal{F}(e)$ are isomorphic to $\mathbb{R}^2$. We can transport features between stalks via the *restriction maps* $\mathcal{F}_{v \trianglelefteq e}$ and $\mathcal{F}_{u \trianglelefteq e}$ and their adjoints $\mathcal{F}^T_{v \trianglelefteq e}$ and $\mathcal{F}^T_{u \trianglelefteq e}$. 17

2.4 Parallel transport on a graph embedded on a sphere. The green vector is transported through the stalks via the path $\mathcal{F}(u) \to \mathcal{F}(v) \to \mathcal{F}(w)$, guided by the appropriate orthogonal restriction maps (connections). Since the green vector at the stalk $\mathcal{F}(u)$ is different from the red vector obtained after parallel transport, we say the transport is *path-dependent*. 21

3.1 Diagram of the sheaf transport weighted by the attention mechanism (with a single attention head). Transporting a vector from stalk $\mathcal{F}(v_1)$ to $\mathcal{F}(v_2)$ is weighted by $\alpha_{v_1 v_2}$, for example. Crucially, the attention weights *from* each node are constrained such that they form a PMF. When the sheaf is trivial, we recover the GAT attention mechanism. . . 30

3.2 General pipeline used in our experiments. First we map the feature vectors to higher-dimensional stalks via an MLP ϕ. Next, we learn attention coefficients and restriction maps. We then use either the SAN or ANSD parameterisation to make predictions and evaluate our models. 37

3.3 Example of the two edge-case synthetic graphs generated, one with the lowest homophily level $h = 0.0$ (a) and the other with the highest homophily level $h = 0.9$ (b). High homophily results in a graph which has very clearly distinguishable classes, whilst low homophily results in a much more visually noisy graph in comparison. 41
3.4 Test accuracy on our synthetic benchmark of 3 classes. Accuracy of $\frac{1}{3}$ is random guessing. The accuracy is plotted as a function of the node homophily level h, which ranges from 0.0 to 0.9 with a step size of 0.1. 42
3.5 Graphs of Decalin and Bicyclopentil molecules, where nodes are carbon atoms and edges are chemical bonds. The node colours show the stable colouring reached by the Weisfeiler-Lehman algorithm. The two molecular graphs are erroneously considered isomorphic by the WL test, although they are not isomorphic. 45

B.1 Spectrum of generated graphs with different homophily levels. As h increases, the classes become more evidently separated. 62

LIST OF TABLES

3.1 Accuracy ± stdev for various node classification datasets. The SAN and ANSD accuracies are reported with varying stalk dimension d, ranging from 2 to 16. These accuracies are to be compared with respect to the baseline GAT accuracy. The top three models for each dataset are coloured by **First**, **Second** and **Third**, respectively. 38

3.2 Accuracy ± stdev for various node classification datasets. Accuracy is reported from 2 to 64 layers increasing in powers of 2. The "best" results corresponds to the number of layers which resulted in the highest accurracy. "OOM" stands for out of memory, whilst "INS" stands for numerical instability. The top three models for each dataset and layer count are coloured by **First**, **Second** and **Third**, respectively. 40

3.3 Accuracy ± stdev for various node classification datasets and models. The datasets are sorted by increasing order of homophily. Our technique is denoted Conn-NSD, while the other Sheaf Diffusion models are Diag-NSD, O(d)-NSD and Gen-NSD. The top three models for each dataset are coloured by **First**, **Second** and **Third**, respectively. The first section includes sheaf-based models, while the second includes other GNN models. 44

3.4 Mean Average Error (MAE) ± stdev on the ZINC regression task for various models. 47

3.5 Training time (in seconds) for 100 epochs on synthetic graphs of different sizes (lower is better). The models are ordered by speed (faster models on the bottom). $O(d)$-NSD is neural sheaf diffusion with orthogonal restriction maps (Bodnar et al., 2022), whilst Conn-NSD is neural sheaf diffusion with pre-computed connection Laplacians (Barbero et al., 2022). For all the models we utilise 1 layer and stalk width $d = 3$ (when applicable). 49

B.1 Hyper-parameter search space for SAN and ANSD. 63

CHAPTER 1

INTRODUCTION

> An expert is a person who has made all the mistakes that can be made in a very narrow field.
>
> *Niels Bohr (1885-1962)*

Traditional Neural Networks are myopic creatures, treating their inputs in isolation. One way to tackle this shortsightedness is through Graph Neural Networks (GNNs) (Scarselli et al., 2008). GNNs have taken the machine learning world by storm, with their main competitive advantage being that they are able to exploit the graph's local neighbourhood structure in the computations. This approach has shown incredible results in many different tasks, ranging from computational drug discovery to social network recommendations.

The Graph Neural Network botanical garden is however not devoid of weeds. Two phenomena are often associated to GNN architectures: oversmoothing and the heterophily problem. The former refers to the fact that building very deep neural networks often leads to poor performance. The latter instead is related to the fact that GNNs are commonly built with the inductive bias that neighbouring nodes are likely to be similar.

Oversmoothing is problematic because important information related to a specific node may be found in nodes which happen to be separated by many edges. GNN layers however operate locally with neighbouring nodes and one would need to stack many layers to be able to gather information from distant interactions. Unfortunately, due to oversmoothing, the signal quickly becomes too smooth to be useful.

Being able to work in heterophilic settings is vital in a wide variety of domains. For example in protein networks, amino acids tend to form links with other amino acids of different types. Another instance is identifying fraudulent behaviour, as fraudster nodes will often try to resemble benign nodes. Ideally a GNN model should be able to function both in heterophilic and homophilic settings.

These issues have been shown to be closely related to the overly-simple graph *topology* which GNNs operate over (Bodnar et al., 2022). Loosely speaking, topological spaces have a notion of neighbourhood, but not distance (metric spaces) or direction (vector spaces). Although graphs are commonly treated as combinatorial structures, they are topological objects as well, with their notion of neighbourhood being naturally defined via the edge structure.

One way in which we can augment the topology over a graph is through the notion of a *sheaf* (Hatcher, 2005). Sheaves are complex objects which give a canonical way of attaching and keeping track of data attached to points over topological spaces. We are interested in a discrete type of sheaf known as a *Cellular Sheaf* (Hansen, 2020; Hansen and Gebhart, 2020; Hansen and Ghrist, 2021), which operates over a discrete topological space known as a Cell Complex (which a graph is a special case of).

A GNN which operates over a Cellular Sheaf is known as a Sheaf Neural Network (SNN) (Bodnar et al., 2022; Hansen and Gebhart, 2020). SNNs have been found both theoretically and empirically to outperform a wide range of GNN models, especially when it comes to the aformentioned problems of oversmoothing and heterophily.

One major breakthrough in machine learning is that of an *attention mechanism* (Vaswani et al., 2017). Attention mechanisms are used to *learn* which objects the neural network should pay the most attention to in the computation. Unsurpisingly, a GNN which computes the convolutions weighted by an attention mechanism, the Graph Attention Network (GAT) (Velickovic et al., 2017), is now a widely popularised technique.

In this thesis, motivated by the great success of attention mechanisms, we embark on an adventurous journey to develop and study attention mechanisms over sheaves. We show that attention-based sheaf neural networks generalise and outperform popular attention-based GNNs and achieve competitive results inline with state-of-the-art GNN models, not only on standard benchmarks but also on ones related designed to measure heterophily and oversmoothing issues.

1.1 Contributions

The contributions of this thesis are the following:

- A systematic overview of Graph Neural Networks, Cellular Sheaf theory and Sheaf Neural Networks. We present Sheaves and Sheaf Neural Networks in a principled way by connecting them directly to traditional GNN literature.
- The formulation of attention mechanisms over cellular sheaves and of the Attentive Sheaf Diffusion (ASD) Partial Differential Equation (PDE). In particular, we show that the ASD PDE is a sheaf generalisation of the Graph Neural Diffusion (GRAND) (Chamberlain et al., 2021) model.
- Sheaf Attention Networks, a direct sheaf generalisation of Graph Attention Networks (GATs) (Velickovic et al., 2017) to cellular sheaves. We show that GATs arise when the underlying sheaf is *trivial*, in the other words with very simple "geometry".
- Attentive Neural Sheaf Diffusion, a diffusion type of neural network which operates over cellular sheaves leveraging an attention mechanism.
- A thorough evaluation of the newly proposed sheaf attention mechanisms. We show that the new models achieve impressive oversmoothing and heterophily performance when compared to state-of-the-art models.

We hope that the contributions of this work spark further interest in sheaf neural networks as it is an incredibly elegant and beautiful area of research, which leverages abstract mathematical concepts and applies them to real-world tasks.

1.2 Outline

Chapter 2 In chapter 2 we discuss the necessary background, which is heavy in both breadth and depth. The chapter starts with an overview of the relevant Graph Neural Network research, modern approaches and issues. Next, we go over Cellular Sheaf theory by building bridges to graph theory when possible to aid with intuition. Finally these two seemingly unrelated topics are tied together via the introduction of Sheaf Neural Networks and their extremely modern literature.

Chapter 3 In chapter 3, we start by formalising an attention mechanism which operates over cellular sheaves. We use this to build an attentive sheaf diffusion partial differential equation. The discretisation of this equation leads to the Sheaf Attention Network (SAN) and Attentive Neural Sheaf Diffusion (ANSD) models. The theoretical

properties of these models are discussed. The models are evaluated on a wide range of benchmarks, both synthetic and real-world.

Chapter 4 In chapter 4, we conclude with a summary of the findings. We importantly also discuss potential exciting future directions with applications to chemistry and other potential computational speed-ups motivated by differential geometry.

CHAPTER 2

BACKGROUND

> Young man, in mathematics you don't understand things. You just get used to them.
>
> *John von Neumann (1903-1957)*

In this chapter we discuss the necessary background, which spans multiple, seemingly unrelated, topics. The aim is to tie them together in a cohesive manner. To set the scene, we start by introducing Graph Neural Networks (GNNs) with popular models which will serve as guidance and intuition. This is followed with a slight detour to the world of cellular sheaf theory, a field of algebraic topology (Hatcher, 2005). Finally, it is then tied together with the introduction of Sheaf Neural Networks (SNNs) (Bodnar et al., 2022; Hansen and Gebhart, 2020), a type of graph neural network which operates over a cellular sheaf.

2.1 Why should we use geometry?

This interlude aims to motivate why we should approach machine learning tasks from a purely geometric perspective in the first place. At a glance, this may be in fact not be so clear. After all, machine learning has flourished as an empirical discipline, breaking records and solving challenging problems in the most diverse of disciplines. What is there to be gained from tackling machine learning from a more geometric perspective?

The Erlangen Programme in 1872 proposed to study geometries through their symmetry transformations, formalised through the use of group theory. This was an effort to unify the many, seemingly unrelated, geometries at the time. The repercussions of this new approach to geometry were profound and not only to mathematics. For example Noether's theorem relates the close relationship between the symmetries of a physical system and its conversation laws. Symmetry is omnipresent in the sciences and, in many ways, defines what we as humans perceive as perfection.

The machine learning world seems to be in a similar situation, with a plethora of architectures which all are seemingly unrelated. In a similar spirit to the Erlangen Programme, Geometric Deep Learning (GDL) (Bronstein et al., 2021, 2017) aims to unify the various deep learning architectures. To do this GDL derives, from first principles, various modern machine learning architectures by examining their group invariant and equivariant properties. Studying machine learning architectures through the lens of the GDL blueprint (Bronstein et al., 2021) allows research to be more guided and systematised.

Consider the canonical task of image classification between cats and dogs. If we shift or rotate the image we expect the classifier to maintain the same prediction, more formally we expect the classifier to be *invariant* to these specific transformations. From a goup theoretic perspective, in this case shifts and rotations of the image signal are considered symmetries of the domain (Bronstein et al., 2021). Applying these transformations does not change the nature of the object.

More formally, consider a signal space Ω, a label space $\mathcal{Y}$ and a group of symmetries $\mathcal{G}$ of the underlying domain. A model $f : \Omega \to \mathcal{Y}$ is called $\mathcal{G}$-invariant if for all $g \in \mathcal{G}$ and $\mathbf{x} \in \Omega$ we have $f(\mathbf{x}) = f(g\mathbf{x})$, in other words the output of the model is unaffacted by the underlying symmetries of the domain.

Remark. By *group* we intend the notion of a group in the algebraic sense (Hamermesh, 2005). For the sake of the discussion it is sufficient to consider this as a set with a group operation which takes two elements in the set and outputs a third element from the set (with some additional convenient properties). The group operation may be thought of as a way to compose two symmetries to retrieve a third symmetry.

Similarly, we have a notion of equivariance. For example, in a segmentation task, we would like the model to modify the output in accordance with the group symmetry g. In other words, a model f is $\mathcal{G}$-equivariant if for all $g \in \mathcal{G}$ and $\mathbf{x} \in \Omega$ we have $f(g\mathbf{x}) = gf(\mathbf{x})$.

Example. A Convolutional Neural Network (CNN) is shift-invariant, however it is not rotation-invariant. It is common to perform data augmentation, artificially adding rotated images to the training set, in hope that the model *learns* the invariance from the data. Learning invariance from the data is however very expensive and difficult, compared to encoding it directly in the first place in the model.

Tackling learning problems from the point of view of invariance and equivariance is an effective way of doing dimensionality reduction. This makes high dimensional problems much more tractable and motivates the effectiveness of a lot of models, such as CNNs, in their specific domains in which they shine. This symmetry ideology is the driving force of the project. We wish to build models which are geometrically grounded to reap the aforementioned benefits which come from this physics-inspired approach.

2.2 Graph Neural Networks

There exist a vast variety of real-world datasets which come with a graph-structure. Consider for example: social networks, protein-interaction networks and control-flow graphs to name a few. The connectivity information which is stored in the form of the edges of a graph is often extremely important and discarding it for a machine learning task often results in a drastic loss in performance. Graph neural networks aim to tackle this problem in a principled way, by generalising neural networks to arbitrarily structured graphs.

Formally, a graph $G = (V, E)$ is a tuple consisting of a set of nodes V and a set of edges E. We can represent each node in the graph with a d-dimensional feature vector $\mathbf{x}_v$ and group all the $n = |V|$ feature vectors into a $n \times d$ matrix $\mathbf{X} = \mathbf{H}^{(0)}$. It is convenient to represent the set of edges E into an $n \times n$ adjacency matrix $\mathbf{A}$. An entry $\mathbf{A}_{ij}$ for nodes v_i and v_j in the adjacency matrix is 1 if there exists an edge $e = v_i \to v_j$ in E and 0 otherwise.

Example. Consider a social network graph. The nodes represent users in the social network and the edges could represent, for instance, a friendship relationship between two users (nodes). A feature vector for a user could include information about how many friends the user has, the time the user has been on the platform and many other different quantities.

Similarly to how neural networks are composed of layers, graph neural networks are composed of of graph neural network layers. The main difference is that while a

neural network takes as input a vector and ouputs a vector, a graph neural network operates over the entire graph, producing (latent) feature vectors for each node.

Definition 2.2.1. The l-th GNN layer $f^{(l)}$ takes as input the graph's (latent) feature matrix at the previous layer $\mathbf{H}^{(l-1)}$ and adjacency matrix $\mathbf{A}$. It outputs a new latent feature matrix $\mathbf{H}^{(l)}$:

$$\mathbf{H}^{(l)} = f^{(l)}\left(\mathbf{H}^{(l-1)}, \mathbf{A}\right) \tag{2.1}$$

In the case of a multi-layer GNN, the first layer $l = 1$, takes as input $\mathbf{H}^{(0)} = \mathbf{X}$, whereas subsequent layers take as input $\mathbf{H}^{(l-1)}$, the latent features produced by the GNN layer immediately before it.

Another way of interpreting equation 2.1 is at a more local scale, with the idea of *message passing*. The intuition of message passing is that many graph neural network models update the representations of each node by exchanging vector information from the neighbouring nodes. More concretely a message passing layer l computes for a node u the latent representation $\mathbf{h}_u^{(l)}$ as follows (Bronstein et al., 2021):

$$\mathbf{h}_u^{(l)} = \phi\left(\mathbf{h}_u^{(l-1)}, \bigoplus_{v \in \mathcal{N}_u} \psi\left(\mathbf{h}_u^{(l-1)}, \mathbf{h}_v^{(l-1)}\right)\right) \tag{2.2}$$

where ϕ is a *message passing function*, ψ is a *readout function* and $\bigoplus$ is a *permutation-invariant aggregation function* (such as $\sum$ or max). In practice ϕ and ψ are usually chosen to be MLPs, but they may have different structures.

The form in equation 2.2 is very interpretable. To update the representation of node u, we consider the neighbourhood and the current representation of node u. We aggregate the neighbourhood information (transformed by ψ) via a permutation invariant function $\bigoplus$ (such as $\sum$ or max). We finally pass the representation of node u alongside the aggregated neighbourhood information to ϕ to compute the final updated representation. The permutation invariant aggregation function guarantees that the output does not depend on how the nodes are ordered.

2.2.1 Graph Convolutional Networks

The task of generalising successful neural network architectures to graphs is an arduous one. Many GNN models have been proposed which operate based on the principle of convolutions, similar to the very successful Convolutional Neural Network (CNN) model. In particular, many of these techniques rely on the concept of a spectral

graph convolution from spectral graph theory (Bruna et al., 2013; Henaff et al., 2015). Possibly the most popular model is the Graph Convolutional Network (GCN) (Kipf and Welling, 2016), whose propagation rule form may be thought of as coming from spectral graph convolutions with some simplifications introduced for computational speedups and accuracy benefits.

Consider the naive propagation rule which implements equation 2.1 the following way:

$$f^{(l)}\left(\mathbf{H}^{(l-1)}, \mathbf{A}\right) = \sigma\left(\mathbf{A}\mathbf{H}^{(l-1)}\mathbf{W}^{(l)}\right) \tag{2.3}$$

where σ is a non-linear activation function (e.g. ReLU) and $\mathbf{W}^{(l)}$ is the weight matrix for layer l with dimension $f_{l-1} \times f_l$ where f_l is the dimension of the feature vector at layer l. This naive model has two main flaws. The calculation $\mathbf{A}\mathbf{H}^{(l-1)}$ results in a new feature matrix in which each feature vector is now the sum of all the nodes in the 1-hop neighbourhood, but importantly not of the node itself (if the adjacency matrix does not contain self-loops). Secondly, depending on the eigenvalues of $\mathbf{A}$, this operation may not preserve scale of the feature vectors and therefore make training the network unstable.

Kipf and Welling (2016) address both of these flaws with the GCN model. The self-loop issue is remedied by artificially introducing self-loops by augmenting the adjacency matrix $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$. The scaling problem is instead addressed via *symmetric normalisation* via $\hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}$, where $\hat{\mathbf{D}}$ is the diagonal node degree matrix of $\hat{\mathbf{A}}$.

Definition 2.2.2. The GCN layer from Kipf and Welling (2016) implements equation 2.1 the following way:

$$\mathbf{H}^{(l)} = \sigma\left(\hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l-1)}\mathbf{W}^{(l)}\right) \tag{2.4}$$

where σ is a non-linear activation function (e.g. ReLU), $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, $\hat{\mathbf{D}}$ is the diagonal node degree matrix of $\hat{\mathbf{A}}$ and $\mathbf{W}^{(l)}$ is a weight matrix.

This update propagation is local (due to the adjacency matrix), meaning that each latent feature vector is updated as a function of its local neighbourhood, weighted by a weight matrix and then symmetrically normalised. This kind of model has proven to be extremely powerful in a myriad of tasks. The weight matrix $\mathbf{W}^{(l)}$ at each layer is learnt from the data through back-propagation, by minimising some loss function (e.g. cross-entropy loss).

Remark. It is perhaps easier to interpret equation 2.4 based on how it updates each individual vector from $\mathbf{h}_i^{(l-1)}$ to $\mathbf{h}_i^{(l)}$, through its message passing form. Consider how it updates a feature vector $\mathbf{h}_i$ for node v_i:

$$\mathbf{h}_i^{(l)} = \sigma\left(\sum_{j \in \mathcal{N}_i} \frac{1}{c_{ij}} \mathbf{h}_j^{(l-1)} \mathbf{W}^{(l)}\right) \tag{2.5}$$

where c_{ij} is the normalisation factor which arises from symmetric normalisation. The sum is over all of the j feature vectors in the 1-hop neighbourhood of v_i (including the feature vector of v_i due to the addition of self-loops). Each feature vector is also updated by the weight matrix $\mathbf{W}^{(l)}$ and then passed through the non-linear activation function σ.

2.2.2 Graph Attention Networks

Attention mechanisms have been responsible for a lot of very significant breakthroughs in machine learning (Bahdanau et al., 2014; Vaswani et al., 2017). They solve a fundamental issue which comes with data of varied size, namely: which part of the data is most relevant to our task. This paradigm-shift has been fundamental for tasks related to natural language, such as learning sentence embeddings (Lin et al., 2017) and machine reading (Cheng et al., 2016). Unsurprisingly, one of the most popular GNN models is one that generalises an attention mechanism to graphs: the Graph Attention Network (GAT) Velickovic et al. (2017).

Remark. Perhaps one of the most important breakthroughs in the machine learning world (especially in natural language processing) has been the proposal of the *Transformer* (Vaswani et al., 2017). The Transformer and GAT models happen to be close cousins. One may in fact imagine a transformer as a graph attention network operating over a fully connected graph (Joshi, 2020).

The core component of the GAT layer is a *shared attentional mechanism* a that performs self-attention on the nodes. $a : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a function which computes raw attention coefficients:

$$e_{ij} = a\left(\mathbf{W}\mathbf{h}_i, \mathbf{W}\mathbf{h}_j\right) \tag{2.6}$$

where e_{ij} measures how much importance the features of node j have towards node i. To preserve the graph structure, *masked attention* is performed, which has the effect of assigning $e_{ij} = 0$ if v_j is not in the neighbourhood of v_i ($v_j \notin \mathcal{N}_i$). The raw attention

coefficients e_{ij} are then turned into a probability mass function (PMF) with a softmax function:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})} \tag{2.7}$$

The form in equation 2.7 is such that $0 \leq \alpha_{ij} \leq 1$ and $\sum_{k \in \mathcal{N}_i} \alpha_{ik} = 1$, meaning that they form a PMF over the edges of node v_i. Setting σ to LeakyReLU and a weight vector $\mathbf{a} \in \mathbb{R}^{2d}$, we retrieve the expanded attention coefficient form from Velickovic et al. (2017).

Definition 2.2.3. The GAT layer from Velickovic et al. (2017) computes attention coefficients as follows:

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^T[\mathbf{W}\mathbf{h}_i || \mathbf{W}\mathbf{h}_j]\right)\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\text{LeakyReLU}\left(\mathbf{a}^T[\mathbf{W}\mathbf{h}_i || \mathbf{W}\mathbf{h}_k]\right)\right)} \tag{2.8}$$

where $\mathbf{a} \in \mathbb{R}^{2d}$ and $\mathbf{W}$ are learnable weights and $||$ denotes vector concatenation.

Remark. Brody et al. (2021) show with their GATv2 model that it is possible to slightly modify the the way that the e_{ij}s are computed in the original GAT model to gain expressive power, maintaining the same computational complexity. The two mechanisms side by side are as follows:

$$\text{GAT (Velickovic et al., 2017): } e_{ij} = \text{LeakyReLU}\left(\mathbf{a}^T[\mathbf{W}\mathbf{h}_i || \mathbf{W}\mathbf{h}_j]\right) \tag{2.9}$$

$$\text{GATv2 (Brody et al., 2021): } e_{ij} = \mathbf{a}^T\left(\text{LeakyReLU}\left(\mathbf{W}[\mathbf{h}_i || \mathbf{h}_j]\right)\right) \tag{2.10}$$

The original GAT attention is limited as the ranking of the attention scores is independent of the query node (called *static attention*). The GATv2 model fixes this short-coming, allowing for *dynamic attention*, which is provably more expressive than its static counterpart (Brody et al., 2021).

Once the attention coefficients are computed, we can finally define how the nodes are updated. Very similarly to how a GCN operates, we simply compute a weighted sum over the neighbours, this time weighted by the attention coefficients:

$$\mathbf{h}_i^{(l)} = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^{(l)} \mathbf{W}^{(l)} \mathbf{h}_j^{(l-1)}\right) \tag{2.11}$$

where $\alpha_{ij}^{(l)}$ is the attention coefficient for the edge $e = v_i \to v_j$ at layer l. Note the close similarity between the GAT and GCN message passing forms. The only difference is that while GCNs compute the coefficient c_{ij} which is a function of the degrees of nodes i and j, GATs *learn* the coefficient α_{ij} via a self attention mechanism. This hints to the fact that GATs are strictly more expressive than GCNs.

Remark. For regularisation purposes, it is common to employ a multi-head attention mechanism, where multiple attention mechanisms are employed in parallel and then aggregated. This takes the following form:

$$\mathbf{h}_i^{(l)} = \Big\Vert_{k=1}^{K} \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^{(l,k)} \mathbf{W}_k^{(l)} \mathbf{h}_j^{(l-1)}\right) \tag{2.12}$$

where $\Big\Vert$ concatenates the K separate attention heads, $\alpha_{ij}^{(l,k)}$ is the attention coefficient of the k-th head at layer l and $\mathbf{W}_k^{(l)}$ is the weight for the k-th head at layer l.

2.2.3 Graph Neural Diffusion

One interesting perspective on GNNs is that they can be treated as a discretisation of a continuous diffusion process. In particular, the Graph Neural Diffusion (GRAND) (Chamberlain et al., 2021) model studies the following (heat) diffusion equation over a graph:

$$\frac{\partial \mathbf{X}(t)}{\partial t} = \boldsymbol{\nabla} \cdot [\mathbf{G}(\mathbf{X}(t), t) \nabla \mathbf{X}(t)] \tag{2.13}$$

where $\boldsymbol{\nabla}\cdot$ and ∇ are (discrete) divergence and gradient operators over a graph respectively and $\mathbf{G}$ is some kind of matrix transformation of the features. The gradient is defined over two nodes connected by an edge such that $(\nabla \mathbf{X}(t))_{ij} = \mathbf{x}_j - \mathbf{x}_i$ and can be thought as a discrete graph version of the standard vector calculus definition. The divergence over a graph instead is defined as the sum of all the neighbouring node features: $(\boldsymbol{\nabla} \cdot \mathbf{X}(t))_i = \sum_{j \in \mathcal{N}_i} \mathbf{x}_j$.

In GRAND, $\mathbf{G}$ is chosen to be an $n \times n$ *attention matrix* $\Lambda(\mathbf{X}(t), t)$, with n being the number of nodes in the graph. We will dissect the precise structure of $\Lambda(\mathbf{X}(t), t)$ in the next chapter, but for now all that is required for the discussion is to know that $\Lambda(\mathbf{X}(t), t)$ is intuitively a way to neatly "group" all the attention coefficients computed as in the GAT model. In particular, the attention matrix is row-stochastic, meaning that each row forms a probability mass function (PMF).

Removing the time-dependence (for simplicity) $\mathbf{G} = \Lambda(\mathbf{X}(t), t) = \Lambda(\mathbf{X}(t))$ in equation 2.13, we get the following linear Partial Differential Equation (PDE):

$$\frac{\partial \mathbf{X}(t)}{\partial t} = (\Lambda - \mathbf{I})\,\mathbf{X}(t) \tag{2.14}$$

This particular PDE is a classical linear PDE and has as solution:

$$\mathbf{X}(t) = \mathbf{X}(0)e^{(\Lambda - \mathbf{I})t} \tag{2.15}$$

where importantly, one can imagine this as a continuous version of the GAT model. This PDE can then be used to compute the latent node features via:

$$\mathbf{X}(T) = \mathbf{X}(0) + \int_0^T \frac{\partial \mathbf{X}(t)}{\partial t} dt \tag{2.16}$$

with initial condition $\mathbf{X}(0) = \phi(\mathbf{X}_{in})$ where ϕ is some transformation of the input node features. The key takeaway here is that many GNN models are *diffusion models* over graphs, which may be thought of as discretisations of some underlying PDE. Experimentally, Chamberlain et al. (2021) show that this model is more robust to oversmoothing.

Remark. This duality between PDEs and discrete models is a similar concept to the popular Neural Ordinary Differential Equations method (Chen et al., 2018). Consider the common residual-type of connection, found for example in Recurrent Neural Networks (Rumelhart et al., 1985):

$$\mathbf{h}_{t+1} = \mathbf{h}_t + f(\mathbf{h}_t, \theta_t) \tag{2.17}$$

We can treat this as a Euler discretisation of the following ordinary differential equation (ODE):

$$\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t, \theta) \tag{2.18}$$

and leverage the arsenal developed during the centuries of mathematics spent studying ODEs to our advantage.

2.2.4 Oversmoothing and Heterophily

Graph Neural Networks are extremely powerful and widespread, yet popular architectures suffer from unideal behaviour under certain conditions. Two of the key

pitfalls we will study and attempt to remediate are: oversmoothing and the heterophily problem.

The issue of *oversmoothing* is a very significant one. In representation learning, there is the general sentiment that *deeper* models with more layers are able to more efficiently learn complicated nested representations. Each layer in some sense is able to learn a more general abstraction. This instead in popular GNN architectures is not the case, in fact stacking too many layers drastically reduces the performance and is related to the signal *smoothing-out* rapidly as more layers are added.

This is also especially problematic for the realm of message-passing GNNs. If two nodes are n-hops away, then one would require a GNN with n layers for the information to reach each-other. Unfortunately as n grows, the signal becomes more and more smooth (even exponentially fast (Di Giovanni et al., 2022)) and the training process is usually dramatically unsuccessful.

Example. Figure 2.1 shows the accuracy on the Cora dataset as more GCN layers are added. As we add more and more layers, we see a sharp decline in the accuracy of the model on the test set due to oversmoothing.

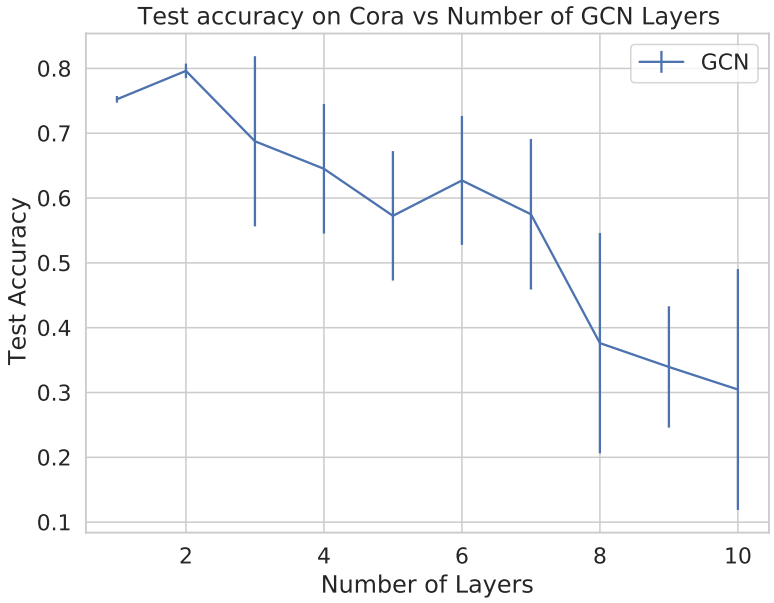


Fig. 2.1 Accuracy on the cora dataset as more GCN layers are added.

Another flaw is what is known as the *heterophily problem*, the fact that GNNs may tend to perform worse in graphs that are more heterophilic. A highly heterophilic (low homophily) graph is one in which neighbouring nodes tend to have different classes. Many GNN models are built with a homophilic inductive bias (GCNs for instance),

meaning that *by design* they tend to function better in homophilic datasets. While this inductive bias is useful in many cases, having a more flexible model would be ideal.

Example. A typical social network may be considered homophilic as users connected to each other are likely to share similar interests. However, consider the task of identifying fraudsters in a bank transaction network. Fraudsters are more likely to connect to accomplices rather than other fradusters, suggesting a more heterophilic relationship (Pandit et al., 2007).

While these two issues at a glance may seem wildly unrelated, it turns out that they are intimately connected (Bodnar et al., 2022; Yan et al., 2021). For instance, through *cellular sheaf theory*, Bodnar et al. (2022) show that both these phenomena can be viewed and addressed from a purely geometric lense. We will discuss in great detail in the next chapter what exactly cellular sheaf theory is. For now the key takeaway is that GNNs are not doomed to these issues if we approach the problem from the right angle.

Example. Figure 2.2 shows the accuracy of a GCN on a synthetic benchmark (see appendix B.2 for details) in which we vary the node homophily level h. The node homophily level $0 \leq h \leq 1$ measures the probability that two nodes are connected if they belong to the same class. We see a trend where the accuracy of the GCN tends to be lower when the node homophily h is lower.

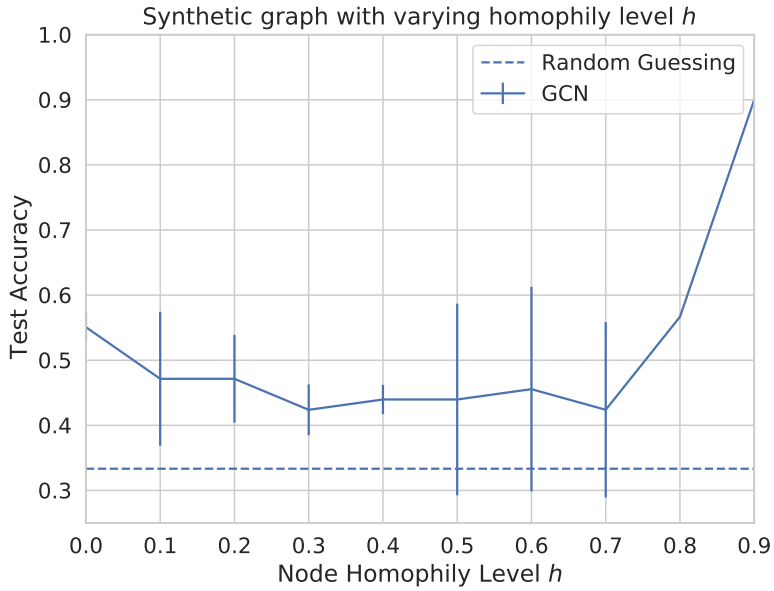


Fig. 2.2 Accuracy of a GCN on a synthetic benchmark with varying node homophily level. Low h means low homophily (high heterophily), high h means high homophily (low heterophily).

Summary

In our journey so far we have encountered popular Graph Neural Network architecures and touched upon the important relationship between PDEs and GNNs, namely that a lot of models arise as discretisations of some underlying PDE. We also discussed pathologies of GNNs such as the oversmoothing and heterophily problems.
We will now take a detour to the world of algebraic topology, to present Cellular Sheaves, machinery which we will need to introduce Sheaf Neural Networks.

2.3 Cellular Sheaf Theory

Graphs are wonderful mathematical objects that allow us to specify pairwise relationships (edges) between its constituent objects (nodes). Crucially, the knowledge of these connection patterns often gives us much more information about the system as a whole. However, in many cases, simply knowing the existence of a relationship discards a lot of information related to the nature of that specific relationship.

Sheaves are a mathematically rigorous solution to this problem, allowing one to attach data to spaces. The theory of sheaves is a very rich and complex one, which roots itself in algebraic topology and algebraic geometry. We will handle a restricted version of this topological beast, namely *cellular sheaves*. The term "cellular" comes from the fact that these types of sheaves are restricted to cell complexes (a type of topological space), although we will be mostly working with graphs, which are a special case of a cell complex (a 1-dimensional cell complex). For further discussion on topology and cellular complexes see appendix A.1.

Definition 2.3.1. A cellular sheaf $(G, \mathcal{F})$ on an *undirected graph* $G = (V, E)$ consists of:

- A vector space $\mathcal{F}(v)$ for each $v \in V$,
- A vector space $\mathcal{F}(e)$ for each $e \in E$,
- A linear map $\mathcal{F}_{v \trianglelefteq e} : \mathcal{F}(v) \to \mathcal{F}(e)$ for each incident node-edge pair $v \trianglelefteq e$.

The vector spaces of the node and edges are called *stalks*, while the linear maps are called *restriction maps*. The sheaf $\mathcal{F}$ bundles the stalks and restriction maps together. The underlying graph specifies the nodes and edges, while the sheaf specifies a network of linear transformations over the underlying nodes and edges (see figure 2.3).

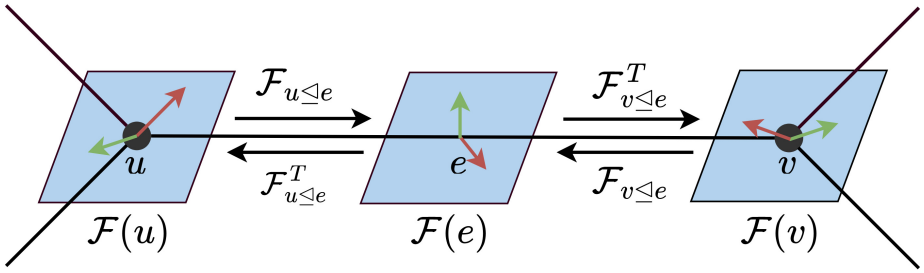


Fig. 2.3 A diagrammatic representation of a sheaf $(G, \mathcal{F})$ for a single edge. The node *stalks* $\mathcal{F}(v)$ and $\mathcal{F}(u)$ and the edge stalk $\mathcal{F}(e)$ are isomorphic to $\mathbb{R}^2$. We can transport features between stalks via the *restriction maps* $\mathcal{F}_{v \trianglelefteq e}$ and $\mathcal{F}_{u \trianglelefteq e}$ and their adjoints $\mathcal{F}^T_{v \trianglelefteq e}$ and $\mathcal{F}^T_{u \trianglelefteq e}$.

A sheaf adds a considerable amount of information to the graph. It is therefore natural to want to compress this into a more manageable form. Formally this is done via *homological algebra*, a sophisticated modern tool which is used to study the compression of information in vector spaces and sequences of linear maps. We will be only scratching the surface, mostly by borrowing notation and terminology.

We start by grouping the node and edge stalks into two different structures. The space which is formed by the node stalks is called the space of 0-cochains, while the space formed by edge stalks is called the space of 1-cochains.

Definition 2.3.2. Given a sheaf $(G, \mathcal{F})$, we define the space of 0-cochains $C^0(G, \mathcal{F})$ as the direct sum over the node stalks $C^0(G, \mathcal{F}) := \bigoplus_{v \in V} \mathcal{F}(v)$. Similarly, the space of 1-cochains $C^1(G, \mathcal{F})$ as the direct sum over the edge stalks $C^1(G, \mathcal{F}) := \bigoplus_{e \in E} \mathcal{F}(e)$.

We call an element $\mathbf{x} \in C^0(G, \mathcal{F})$ a 0-cochain. We can interpret it as a choice of data $\mathbf{x}_v \in \mathcal{F}(v)$ for every node in the graph, where $\mathbf{x}$ is the concatenation of the individual vectors $\mathbf{x}_v$ in all the node stalks $\mathcal{F}(v)$. Similarly, a 1-cochain $\mathbf{y} \in C^1(G, \mathcal{F})$ is a choice of data for every edge in the graph. Building the spaces $C^0(G, \mathcal{F})$ and $C^1(G, \mathcal{F})$ allows us to construct a linear *co-boundary map* $\delta : C^0(G, \mathcal{F}) \to C^1(G, \mathcal{F})$ from 0-cochains to 1-cochains. From an opinion dynamics perspective (Hansen and Ghrist, 2021), the node stalks may be thought of as the private space of opinions and the edge stalks as the space in which these opinions are shared in a public discourse space. The co-boundary map δ then measures the disagreement between all the nodes over the edge spaces, which will depend on the restriction maps.

Definition 2.3.3. Given some arbitrary orientation for each edge $e = u \to v \in E$, we define the co-boundary map $\delta : C^0(G, \mathcal{F}) \to C^1(G, \mathcal{F})$ as $\delta(\mathbf{x})_e = \mathcal{F}_{v \trianglelefteq e} \mathbf{x}_v - \mathcal{F}_{u \trianglelefteq e} \mathbf{x}_u$.

Here $\mathbf{x} \in C^0(G, \mathcal{F})$ is a 0-cochain and $\mathbf{x}_v \in \mathcal{F}(v)$ is the vector of $\mathbf{x}$ at the node stalk $\mathcal{F}(v)$.

The choice of orientation is not important as we are fundamentally interested in the kernel of δ, which is invariant under this choice. Continuing with the opinions dynamic analogy, if δ measures the total disagreement between nodes, then the kernel of δ gives us the subspace of 0-cochains that perfectly "agree" over the edges.

Remark. The kernel of a linear map is the linear subspace which contains all of elements in the domain which are mapped to the zero vector. If a 0-cochain $\mathbf{x}$ is in the kernel of δ, then $\delta\mathbf{x} = 0$ and this is only the case if for every edge $e = u \to v \in E$, $\delta(\mathbf{x})_e = \mathcal{F}_{v \trianglelefteq e}\mathbf{x}_v - \mathcal{F}_{u \trianglelefteq e}\mathbf{x}_u = 0$ imposing $\mathcal{F}_{v \trianglelefteq e}\mathbf{x}_v = \mathcal{F}_{u \trianglelefteq e}\mathbf{x}_u$. In other words, all of the nodes agree over their respective edge discourse space.

If one considers the sheaf as inducing a global constraint satisfaction problem, then an element of the kernel of δ is a solution. The set of solutions forms a vector space known as the *zeroth cohomology* or the space of *global sections*.

Definition 2.3.4. Given a sheaf $(G, \mathcal{F})$, the zeroth cohomology or the space of global sections of $\mathcal{F}$ is defined as:

$$H^0(G, \mathcal{F}) = \ker \delta \subset C^0(G, \mathcal{F}) \tag{2.19}$$

2.3.1 Sheaf Laplacians

Having discussed the basic machinery that surrounds cellular sheaves, we can now introduce the concept of a sheaf Laplacian. We introduce this object by relating it to the graph Laplacian.

Given an incidence matrix $\mathbf{B}$, the *graph Laplacian* $\mathbf{L}$ is given by $\mathbf{L} = \mathbf{B}\mathbf{B}^T$. This object may be viewed as a matrix-form of the discretisation of the well-known Laplacian, named after the French mathematician Pierre-Simon de Laplace (1749–1827). We now wish to construct the *sheaf Laplacian*, which generalises the idea of a graph Laplacian to cellular sheaves. This extension is straightforward if one considers δ as a sheaf generalisation of the transposed incidence matrix $\mathbf{B}^T$.

Remark. The element $\mathbf{B}_{ve}$ of the $|V| \times |E|$ incidence matrix for a vertex $v \in V$ and edge $e = v_i \to v_j \in E$ has a value of 1 if $v = v_i$, -1 if $v = v_j$ and 0 otherwise. For a sheaf with stalks isomorphic to $\mathbb{R}$ and identity restriction maps (a trivial sheaf), then we have that $\delta = \mathbf{B}^T$. Removing the restrictions on stalk dimension and identity restriction maps, δ acts as a block-matrix which generalises $\mathbf{B}^T$ to sheaves.

Definition 2.3.5. The sheaf Laplacian of a sheaf is a map $\mathbf{L}_{\mathcal{F}} : C^0(G,\mathcal{F}) \to C^0(G,\mathcal{F})$ defined as $\mathbf{L}_{\mathcal{F}} = \delta^\top \delta$.

The sheaf Laplacian is a symmetric positive semi-definite (by construction) block matrix. The diagonal blocks are $\mathbf{L}_{\mathcal{F}_{v,v}} = \sum_{v \trianglelefteq e} \mathcal{F}^\top_{v \trianglelefteq e} \mathcal{F}_{v \trianglelefteq e}$, while the off-diagonal blocks are $\mathbf{L}_{\mathcal{F}_{v,u}} = -\mathcal{F}^\top_{v \trianglelefteq e} \mathcal{F}_{u \trianglelefteq e}$.

$$\mathbf{L}_{\mathcal{F}} = \begin{bmatrix} \sum_{v_1 \trianglelefteq e} \mathcal{F}^T_{v_1 \trianglelefteq e} \mathcal{F}_{v_1 \trianglelefteq e} & \dots & -\mathcal{F}^T_{v_1 \trianglelefteq e} \mathcal{F}_{v_n \trianglelefteq e} \\ \vdots & \ddots & \vdots \\ -\mathcal{F}^T_{v_n \trianglelefteq e} \mathcal{F}_{v_1 \trianglelefteq e} & \dots & \sum_{v_n \trianglelefteq e} \mathcal{F}^T_{v_n \trianglelefteq e} \mathcal{F}_{v_n \trianglelefteq e} \end{bmatrix} \tag{2.20}$$

A particularly interesting edge-case for the sheaf Laplacian is when all of the stalks are 1-dimensional and the restriction maps are identity maps. We call such a sheaf *trivial*.

Definition 2.3.6. A sheaf $(G,\mathcal{F})$ is called *trivial* when the stalks are all of dimension 1 and the restriction maps are all identity maps such that $\mathcal{F}_{v \trianglelefteq e} = 1$.

Lemma 2.3.7. When a sheaf $(G,\mathcal{F})$ is trivial, then the sheaf Laplacian $L_{\mathcal{F}}$ becomes the well-known graph Laplacian L_G.

Proof. Observe the matrix structure in equation 2.20 and consider a trivial sheaf. With a trivial sheaf, the diagonal entry $\mathbf{L}_{\mathcal{F}_{v,v}}$ will count how many 1-hop neighbours node v has. This is because $\mathcal{F}^T_{v_1 \trianglelefteq e} \mathcal{F}_{v_1 \trianglelefteq e} = 1$ if the edge exists and 0 otherwise and we are summing over the neighbourhood of the node. Instead, the off-diagonal entry $\mathbf{L}_{\mathcal{F}_{v,u}}$ will be -1 when there exists an edge $e = v \to u \in E$ and 0 otherwise. This is precisely the structure of the graph Laplacian, which can be thought of as $\mathbf{L}_G = \mathbf{D}_G - \mathbf{A}_G$, where $\mathbf{D}_G$ is the diagonal degree matrix and $\mathbf{A}_G$ is the adjacency matrix. □

Lemma 2.3.7 is intuitively extremely important and acts as a bridge between (spectral) graph theory and (spectral) cellular sheaf theory. In essence it tells us that sheaf Laplacians generalise graph Laplacians.

It is common to normalise the Laplacian matrix to bound its spectrum, giving rise to the normalised sheaf Laplacian $\Delta_{\mathcal{F}}$.

Definition 2.3.8. The *normalised sheaf Laplacian* $\Delta_{\mathcal{F}}$ is defined as $\Delta_{\mathcal{F}} = \mathbf{D}^{-\frac{1}{2}} \mathbf{L}_{\mathcal{F}} \mathbf{D}^{-\frac{1}{2}}$ where D is the block-diagonal of $\mathbf{L}_{\mathcal{F}}$.

Harmonic signals $\mathbf{x} \in C^0(G, \mathcal{F})$ are signals which perfectly agree with the structure of the sheaf Laplacian, such that $\mathbf{L}_\mathcal{F}\mathbf{x} = 0$. There is a strong connection between the signals which agree with the sheaf Laplacian $\mathbf{L}_\mathcal{F}$ and signals which agree with the co-boundary operator δ (global sections). In fact, the central theorem of discrete Hodge theory states that the space of harmonic signals coincides with the global sections of the sheaf, in other words that $\ker \mathbf{L}_\mathcal{F} = H^0(G, \mathcal{F})$.

Theorem 2.3.9. (Hodge theorem) Given a sheaf $(G, \mathcal{F})$, $\ker(\mathbf{L}_\mathcal{F}) = H^0(\mathcal{G}, \mathcal{F})$.

Proof. Recall that by definition $\mathbf{L}_\mathcal{F} = \delta^\top \delta$. Thus $\ker \mathbf{L}_\mathcal{F} = \ker \delta^\top \delta = \ker \delta = H^0(\mathcal{G}, \mathcal{F})$, where the last step is by definition of the global sections of a sheaf. □

The proof of the Hodge theorem seems almost trivial when working with sheaves over graphs, but stems from much deeper results over cell complexes. This result is useful in practice as it allows us to understand the kernel of the sheaf Laplacian as elements which satisfy the constraint satisfaction problem induced by δ.

2.3.2 $O(d)$-bundles and Connection Laplacians

In our work, we make the simplification that all the stalks are of dimension d, meaning that all of the restriction maps are $d \times d$ matrices. It is often interesting to impose restriction on the restriction maps, for example it is particularly convenient to restrict the maps to be orthogonal, or more formally part of the orthogonal Lie group $O(d)$.

Definition 2.3.10. The *orthogonal (Lie) group* of dimension d, denoted $O(d)$, is the group of $d \times d$ orthogonal matrices together with matrix multiplication.

Example. A matrix $\mathbf{Q} \in O(d)$ is such that $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$ and is guaranteed to have a determinant of ± 1. Orthogonal matrices can be thought of as rotations as they have a unit determinant. For instance all elements of $O(2)$ may be written as:

$$\begin{bmatrix} -\cos(\theta) & \sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{2.21}$$

The remarkable observation is that while this matrix has 4 entries, it actually only has 1 free parameter in the form of the rotation angle θ. It is worth mentioning that if we restrict the determinant to be $+1$ (not ± 1), then we retrieve the *special* orthogonal group $SO(d)$.

When all the restriction maps are part of the orthogonal group, ie. $\mathcal{F}_{v \trianglelefteq e} \in O(d)$, then the sheaf is called a discrete $O(d)$-bundle, a discrete analogy of vector bundles from differential geometry (Tu, 2011). A vector bundle is a precise formulation of how to parameterise a family of vector spaces over points on a manifold. A *connection* over a vector bundle allows one to define the notion of *parallel transport* on the bundle. Parallel transport is a mechanism to "connect" nearby vector spaces over the manifold.

In our discretisation, the graph acts as the manifold and the sheaf Laplacian then specifies the way in which the nodes are connected together, with the restriction maps. Since the maps are all orthogonal, the vectors are rotated across the graph (see figure 2.4). For this reason, the sheaf Laplacian in the case of $O(d)$ restriction maps is also called the *connection Laplacian* (Singer and Wu, 2012), due to its relationship with connections and parallel transport.

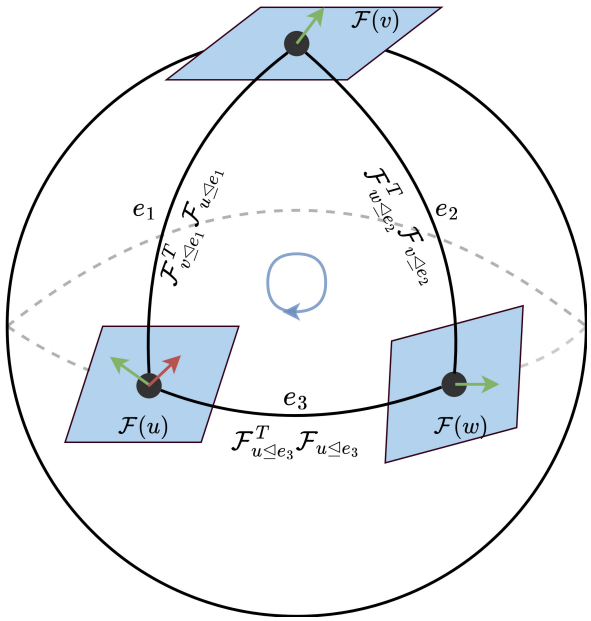


Fig. 2.4 Parallel transport on a graph embedded on a sphere. The green vector is transported through the stalks via the path $\mathcal{F}(u) \to \mathcal{F}(v) \to \mathcal{F}(w)$, guided by the appropriate orthogonal restriction maps (connections). Since the green vector at the stalk $\mathcal{F}(u)$ is different from the red vector obtained after parallel transport, we say the transport is *path-dependent*.

Orthogonal restriction maps are advantageous because orthogonal matrices have fewer free parameters, making them more efficient to work with. The Lie group $O(d)$ has a $d(d-1)/2$-dimensional manifold structure (compared to the d^2-dimensional general linear group describing all invertible matrices). When $d = 2$, for instance, 2×2 rotation matrices have only one free parameter (the rotation angle).

Summary

We introduced cellular sheaves and motivated their introduction by relating the sheaf Laplacian to the graph Laplacian. We further discussed how imposing the restriction maps to be orthogonal relates sheaves to vector bundles. We are now finally ready to introduce the notion of a *Sheaf Neural Network*.

2.4 Sheaf Diffusion

Having defined a sheaf Laplacian, it is natural to study the linear dynamical system which it induces. We will later discretise this dynamical system to retrieve the notion of a Sheaf Neural Network.

Definition 2.4.1. For $\alpha > 0$ and a 0-cochain $\mathbf{x} \in C^0(\mathcal{G}, \mathcal{F})$ we define the *sheaf heat equation*:

$$\frac{\partial \mathbf{x}(t)}{\partial t} = -\alpha \mathbf{L}_{\mathcal{F}} \mathbf{x}(t) \tag{2.22}$$

The dynamics of the diffusion of the sheaf heat equation tend to push the opinion of an individual $\mathbf{x}_v(t) \in \mathcal{F}(v)$ to increase in "agreement" with its neighbours as t increases. As $t \to \infty$ this agreement becomes exact and the original signal $\mathbf{x}(0)$ is projected onto the harmonic space, becoming a harmonic signal. This agreement is dependent on the structure of the sheaf and, as a consequence, the sheaf Laplacian.

Theorem 2.4.2. (Hansen and Ghrist, 2021) Solutions $\mathbf{x}(t)$ of the sheaf heat equation (2.22), in the limit $t \to \infty$ converge to the orthogonal projection of the starting signal $\mathbf{x}(0)$ onto $H^0(\mathcal{G}, \mathcal{F})$.

Proof. See appendix A.2. □

This result implies that, in the time limit, sheaf diffusion can be seen as a global synchronization process. Recall that the 0-cochains which are in the space of global sections $H^0(\mathcal{G}, \mathcal{F})$ are those signals that are solutions to the global constraint satisfaction problem induced by the sheaf. As $t \to \infty$ we orthogonally project the starting signal $\mathbf{x}(0)$ onto the space of global sections. Note that if $H^0(\mathcal{G}, \mathcal{F})$ is trivial, meaning that only the 0 vector is in the space of global sections, then all $\mathbf{x}(0)$ will converge to 0. This suggests that having a non-trivial kernel is more desirable and interesting.

Remark. In particular, the solution to equation 2.22 is:

$$\mathbf{x}(t) = \exp(-t\alpha\mathbf{L}_{\mathcal{F}})\mathbf{x}(0) \tag{2.23}$$

meaning that this process converges exponentially quickly to the orthogonal projection onto $H^0(\mathcal{G},\mathcal{F})$. The rate of convergence is related to the spectrum of the sheaf Laplacian (Hansen and Ghrist, 2019).

2.4.1 Neural Sheaf Diffusion

Consider a graph $G = (V,E)$ where each node $v \in V$ has a d-dimensional feature vector $\mathbf{x}_v \in \mathcal{F}(v)$. We construct an nd-dimensional vector $\mathbf{x} \in C^0(G,\mathcal{F})$ by column-stacking the individual vectors $\mathbf{x}_v$. Allowing for f feature channels, we produce the feature matrix $\mathbf{X} \in \mathbb{R}^{(nd)\times f}$. The columns of $\mathbf{X}$ are vectors in $C^0(G,\mathcal{F})$, one for each of the f channels. Bodnar et al. (2022) study a sheaf diffusion equation similar to equation 2.22:

Definition 2.4.3. We define the *normalised* sheaf heat equation over graph features:

$$\mathbf{X}(0) = \mathbf{X}, \quad \frac{\partial}{\partial t}\mathbf{X}(t) = -\Delta_{\mathcal{F}}\mathbf{X}(t) \tag{2.24}$$

where $\mathbf{X} \in \mathbb{R}^{(nd)\times f}$ is the graph's feature matrix and $\Delta_{\mathcal{F}}$ is the normalised sheaf Laplacian.

The partial differential equation (PDE) in equation 2.4.1 has as initial condition $\mathbf{X}(0)$ the feature matrix $\mathbf{X} \in \mathbb{R}^{(nd)\times f}$. The same convergence result in theorem 2.4.2 holds as well for this particular PDE up to a $D^{-\frac{1}{2}}$ normalisation.

Consider an Euler discretisation with unit time step ($\tau = 1$) of equation :

$$\frac{\mathbf{X}(t+1) - \mathbf{X}(t)}{\tau} = -\Delta_{\mathcal{F}}\mathbf{X}(t) \tag{2.25}$$

$$\mathbf{X}(t+1) = \mathbf{X}(t) - \Delta_{\mathcal{F}}\mathbf{X}(t) \tag{2.26}$$

$$\mathbf{X}(t+1) = (\mathbf{I} - \Delta_{\mathcal{F}})\mathbf{X}(t) \tag{2.27}$$

We can equip equation 2.27 with weight matrices $\mathbf{W}_1 \in \mathbb{R}^{d\times d}$ and $\mathbf{W}_2 \in \mathbb{R}^{f_1\times f_2}$ and a non-linearity σ to derive the *Sheaf Convolutional Network* (SCN) model proposed by Hansen and Gebhart (2020).

Definition 2.4.4. The Sheaf Convolutional Network (SCN) model (Hansen and Gebhart, 2020) takes the form:

$$\hat{\mathbf{X}} = \sigma\left(\left(\mathbf{I}_{nd} - \Delta_{\mathcal{F}}\right)\left(\mathbf{I}_n \otimes \mathbf{W}_1\right)\mathbf{X}_t\mathbf{W}_2\right) \tag{2.28}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d\times d}$ and $\mathbf{W}_2 \in \mathbb{R}^{f_1\times f_2}$ are weight matrices, σ is a non-linear activation function, f_1 and f_2 are the input and output channels respectively, and $\otimes$ is the Kronecker product.

It is natural to call this model a Sheaf Convolutional Network as when the sheaf is trivial we recover the Graph Convolutional Network model.

Theorem 2.4.5. The Graph Convolutional Network (GCN) is a special case of the Sheaf Convolutional Network (SCN), which arises when the underlying sheaf $(G,\mathcal{F})$ is trivial.

Proof. Using $\mathbf{L}_{\mathcal{F}} = \mathbf{D}_{\mathcal{F}} - \mathbf{A}_{\mathcal{F}}$, we can re-write the normalised sheaf Laplacian as $\Delta_{\mathcal{F}} = \mathbf{D}^{-\frac{1}{2}}L_{\mathcal{F}}\mathbf{D}^{-\frac{1}{2}} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{D}_{\mathcal{F}} - \mathbf{A}_{\mathcal{F}})\mathbf{D}^{-\frac{1}{2}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}_{\mathcal{F}}\mathbf{D}^{-\frac{1}{2}}$ and substitute that in the SCN equation:

$$\begin{aligned}
\hat{\mathbf{X}} &= \sigma\left(\left(\mathbf{I}_{nd} - \Delta_{\mathcal{F}}\right)\left(\mathbf{I}_n \otimes \mathbf{W}_1\right)\mathbf{X}\mathbf{W}_2\right) && (2.29)\\
&= \sigma\left(\left(\mathbf{I}_{nd} - \left(\mathbf{I}_{nd} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}_{\mathcal{F}}\mathbf{D}^{-\frac{1}{2}}\right)\right)\left(\mathbf{I}_n \otimes \mathbf{W}_1\right)\mathbf{X}\mathbf{W}_2\right) && (2.30)\\
&= \sigma\left(\mathbf{D}^{-\frac{1}{2}}\mathbf{A}_{\mathcal{F}}\mathbf{D}^{-\frac{1}{2}}\left(\mathbf{I}_n \otimes \mathbf{W}_1\right)\mathbf{X}\mathbf{W}_2\right) && (2.31)
\end{aligned}$$

When the sheaf $(G,\mathcal{F})$ is trivial then, with lemma 2.3.7, we know that the sheaf Laplacian $\mathbf{L}_{\mathcal{F}}$ is the graph Laplacian $\mathbf{L}_G$. On top of this, the adjacency matrices and diagonal degree matrices match, ie. $\mathbf{D}_{\mathcal{F}} = \mathbf{D}_G$ and $\mathbf{A}_{\mathcal{F}} = \mathbf{A}_G$. Furthermore, when the sheaf is trivial (d=1), $\mathbf{W}_1 \in \mathbb{R}^{d\times d}$ becomes a scalar factor and can be dropped. We therefore retrieve the GCN equation:

$$\begin{aligned}
\hat{\mathbf{X}} &= \sigma\left(\mathbf{D}^{-\frac{1}{2}}\mathbf{A}_{\mathcal{F}}\mathbf{D}^{-\frac{1}{2}}\left(\mathbf{I}_n \otimes \mathbf{W}_1\right)\mathbf{X}\mathbf{W}_2\right) && (2.32)\\
&= \sigma\left(\mathbf{D}^{-\frac{1}{2}}\mathbf{A}_G\mathbf{D}^{-\frac{1}{2}}\mathbf{X}\mathbf{W}_2\right) && (2.33)
\end{aligned}$$

□

While interesting from a theoretical perspective, the SCN model in equation 2.27 from Hansen and Gebhart (2020) is not practical as it assumes that the sheaf is known a-priori. Bodnar et al. (2022) instead propose a *Neural Sheaf Diffusion* (NSD) model with a sheaf $\mathcal{F}(t)$ which now *evolves over time*. The model from Bodnar et al. (2022) takes the form:

$$\mathbf{X}_{t+1} = \mathbf{X}_t - \sigma\left(\Delta_{\mathcal{F}(t)}\left(\mathbf{I}_n \otimes \mathbf{W}_1^t\right)\mathbf{X}_t\mathbf{W}_2^t\right) \tag{2.34}$$

with a crucial difference being that now the $d \times d$ restriction maps $\mathcal{F}_{v \trianglelefteq e:=(v,u)}$ are *learnt* via a parametric function $\Phi : \mathbb{R}^{d\times 2} \to \mathbb{R}^{d\times d}$ such that $\mathcal{F}_{v \trianglelefteq e:=(v,u)} = \Phi(\mathbf{x}_v, \mathbf{x}_u)$. In practice, $\Phi(\mathbf{x}_v, \mathbf{x}_u) = \sigma\left(\mathbf{W}[\mathbf{x}_u || \mathbf{x}_v]\right)$. The output vector is then reshaped into a matrix, potentially enforcing specific structure of the restriction maps (orthogonality for example). It is crucial that the function Φ is asymmetric, allowing to learn asymmetric restriction maps as this gives the model more expressivity and modelling capacity.

Fixing the class of restriction maps allows for different separation properties. In the most general case, one can learn general linear maps. This is the most powerful setting, but most prone to overfitting and harder to train due to the spectrum being potentially unbounded. Strictly diagonal restriction maps reduce the number of learnt parameters, but have the limitation that the stalk dimensions now cannot mix (as the non-diagonal terms are all 0).

A convenient middle ground is when the maps are orthogonal, giving rise to a discrete $O(d)$-vector bundle. Since all the eigenvalues of orthogonal maps are ± 1, stacking multiple layers does not result in exploding or vanishing gradients. Furthermore, Bodnar et al. (2022) prove that orthogonal maps are the most efficient, in terms of separation power, when it comes to stalk width d.

Remark. While Bodnar et al. (2022) suggest to learn the sheaf with a parametric function Φ, which is fitted with standard gradient-based methods, Barbero et al. (2022) instead propose to directly compute the (orthogonal) restriction maps from the data basing themselves in Riemannian geometry. The graph is treated as a manifold and the restriction maps are then constructed to optimally align the tangent space of neighbouring nodes in the graph. This technique is computationally advantageous, at the cost of removing the added flexibility which comes from parameterising the restriction maps with Φ.

Sheaf neural networks are particularly interesting because they theoretically and empirically are geared to tackle issues related to oversmoothing and heterophily (Bodnar

et al., 2022). It is for this very reason that we wish to expand upon the current literature on SNNs: Sheaf Neural Networks are both elegant theoretically, and have shown very promising empirical results.

Summary

We started the chapter by discussing different relevant types of GNNs and some of their pitfalls. We then took a detour to the world of Cellular Sheaf theory to introduce the required machinery for our model of interest. We brought all of the concepts together to introduce the notion of Sheaf Neural Networks. We are now well-equipped to tackle the goal of the project: developing attention-based sheaf neural networks.

CHAPTER 3

ATTENTION MECHANISMS OVER SHEAVES

> There are two possible outcomes: if the result confirms the hypothesis, then you've made a measurement. If the result is contrary to the hypothesis, then you've made a discovery.
>
> *Enrico Fermi (1901-1954)*

In this chapter we set off with the ambitious goal of developing an attention mechanism over cellular sheaves. One of the main challenges consists in motivating the chosen structure of the attention mechanism. To do this in a principled manner, we construct an attentive sheaf diffusion PDE and show how it relates to existing GNN literature.

The discretisation of the aformentioned PDE leads to two models of interest: the Sheaf Attention Network (SAN) and Attentive Neural Sheaf Diffusion (ANSD). We prove interesting theoretical properties of these models and empirically show that they address issues of oversmoothing and heterophily present in, for example, GANs. The motivation is that, similarly to neural sheaf diffusion, we can leverage the more complex sheaf geometry to create more expressive attention-based models.

3.1 Attentive Sheaf Diffusion

We start by defining the notion of an *attention matrix* $\Lambda \odot \mathbf{A}_G$ over a graph, which we then generalise to a sheaf with arbitrary stalk dimension d. An attention matrix has two fundamental properties, first each entry is greater than 0 and second that each row sums to 1, making it a row-wise stochastic matrix. This is simply a tool which we can use to summarise the attention information into a single algebraic structure.

Another desire is that our attention matrix respects the edge structure of the graph. This is done with the operation $\odot\mathbf{A}_G$, where $\odot$ represents element-wise multiplication. Recall the graph adjacency matrix $\mathbf{A}_G$ has value for entry uv: 1 if the edge $e = u \to v$ exists and 0 otherwise (including potentially self-loops). This operation therefore has the effect of setting all entries in the attention matrix which do not respect the graph structure to 0 and leaving the rest untouched.

Example. Consider a (directed) graph with 3 nodes a, b and c with edges $a \to b$ and $b \to c$ (adding self-loops). The operation $\Lambda \odot \mathbf{A}_G$ then has the structure (denoting entry uv of Λ by α_{uv}:

$$\Lambda \odot \mathbf{A}_G = \begin{bmatrix} \alpha_{aa} & \alpha_{ab} & \alpha_{ac} \\ \alpha_{ba} & \alpha_{bb} & \alpha_{bc} \\ \alpha_{ca} & \alpha_{cb} & \alpha_{cc} \end{bmatrix} \odot \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \alpha_{aa} & \alpha_{ab} & 0 \\ 0 & \alpha_{bb} & \alpha_{bc} \\ 0 & 0 & \alpha_{cc} \end{bmatrix} \tag{3.1}$$

Crucially, we have the following row-stochasticity constraints: $\alpha_{aa} + \alpha_{ab} = 1$, $\alpha_{bb} + \alpha_{bc} = 1$, and $\alpha_{cc} = 1$ and that every entry uv is positive $\alpha_{uv} \geq 0$. Note that the positivity constraints alongside the row-sum-to-one constraints also mean that all $\alpha_{uv} \leq 1$.

The structure of the attention matrix follows how the attention matrix is computed for the GAT model, where we make the tacit assumption that the sheaf is trivial. To generalise it to d-dimensional stalks we introduce the following structure $\Lambda \otimes \mathbf{1}_d \odot \mathbf{A}_{\mathcal{F}}$ where $\otimes$ is the Kronecker product and $\mathbf{1}_d$ is a $d \times d$ matrix with each entry 1. We also promote the graph adjacency matrix to the sheaf adjacency matrix, where now the entries are the restriction maps connecting two neighbouring nodes.

Example. The operation $\Lambda \otimes \mathbf{1}_d \odot \mathbf{A}_{\mathcal{F}}$ remains intuitive. Consider the same (directed) graph as before with nodes a, b and c and edges $e_1 = a \to b$ and $e_2 = b \to c$ (with self loops). This time remaining general with d-dimensional stalks. The result of $\Lambda \otimes \mathbf{1}_d$ is a block matrix where now the entry Λ_{uv} becomes a $d \times d$ matrix with each element being α_{uv}. We denote this $d \times d$ matrix with all entries being α_{uv} as $\boldsymbol{\alpha}_{uv}$, in other words $\boldsymbol{\alpha}_{uv} = \alpha_{uv}\mathbf{1}_d$.

$$\Lambda \otimes \mathbf{1}_d \odot \mathbf{A}_{\mathcal{F}} = \begin{bmatrix} \alpha_{aa} & \alpha_{ab} & \alpha_{ac} \\ \alpha_{ba} & \alpha_{bb} & \alpha_{bc} \\ \alpha_{ca} & \alpha_{cb} & \alpha_{cc} \end{bmatrix} \otimes \mathbf{1}_d \odot \begin{bmatrix} \mathcal{F}_{a \trianglelefteq e_a}^T \mathcal{F}_{a \trianglelefteq e_a} & \mathcal{F}_{b \trianglelefteq e_1}^T \mathcal{F}_{a \trianglelefteq e_1} & \mathbf{0}_d \\ \mathbf{0}_d & \mathcal{F}_{b \trianglelefteq e_b}^T \mathcal{F}_{b \trianglelefteq e_b} & \mathcal{F}_{c \trianglelefteq e_2}^T \mathcal{F}_{b \trianglelefteq e_2} \\ \mathbf{0}_d & \mathbf{0}_d & \mathcal{F}_{c \trianglelefteq e_c}^T \mathcal{F}_{a \trianglelefteq e_c} \end{bmatrix} \tag{3.2}$$

$$= \begin{bmatrix} \boldsymbol{\alpha}_{aa} & \boldsymbol{\alpha}_{ab} & \boldsymbol{\alpha}_{ac} \\ \boldsymbol{\alpha}_{ba} & \boldsymbol{\alpha}_{bb} & \boldsymbol{\alpha}_{bc} \\ \boldsymbol{\alpha}_{ca} & \boldsymbol{\alpha}_{cb} & \boldsymbol{\alpha}_{cc} \end{bmatrix} \odot \begin{bmatrix} \mathcal{F}_{a \trianglelefteq e_a}^T \mathcal{F}_{a \trianglelefteq e_a} & \mathcal{F}_{b \trianglelefteq e_1}^T \mathcal{F}_{a \trianglelefteq e_1} & \mathbf{0}_d \\ \mathbf{0}_d & \mathcal{F}_{b \trianglelefteq e_b}^T \mathcal{F}_{b \trianglelefteq e_b} & \mathcal{F}_{c \trianglelefteq e_2}^T \mathcal{F}_{b \trianglelefteq e_2} \\ \mathbf{0}_d & \mathbf{0}_d & \mathcal{F}_{c \trianglelefteq e_c}^T \mathcal{F}_{a \trianglelefteq e_c} \end{bmatrix} \tag{3.3}$$

$$= \begin{bmatrix} \alpha_{aa} \mathcal{F}_{a \trianglelefteq e_a}^T \mathcal{F}_{a \trianglelefteq e_a} & \alpha_{ab} \mathcal{F}_{b \trianglelefteq e_1}^T \mathcal{F}_{a \trianglelefteq e_1} & \mathbf{0}_d \\ \mathbf{0}_d & \alpha_{bb} \mathcal{F}_{b \trianglelefteq e_b}^T \mathcal{F}_{b \trianglelefteq e_b} & \alpha_{bc} \mathcal{F}_{c \trianglelefteq e_2}^T \mathcal{F}_{b \trianglelefteq e_2} \\ \mathbf{0}_d & \mathbf{0}_d & \alpha_{cc} \mathcal{F}_{c \trianglelefteq e_c}^T \mathcal{F}_{a \trianglelefteq e_c} \end{bmatrix} \tag{3.4}$$

$$\tag{3.5}$$

where in the last step we scalar multiply by α_{uv} instead of $\boldsymbol{\alpha}_{uv}$ as we are effectively multiplying each element of the adjacency matrix entry by α_{uv} as all the elements of $\boldsymbol{\alpha}_{uv}$ are α_{uv}. The same row-stochasticity constraints on the attention coefficients apply as before.

When $d = 1$ we recover the original trivial sheaf operation described in the previous section. When d is arbitrary, the attention coefficient structure is the same, but they now multiply $d \times d$ restriction maps instead of scalars.

The way through which we compute the attention coefficients α_{uv} follows exactly how it is done in GAT, although when the stalks are higher dimensional, we input all the feature channels instead of the feature vector. In our experiments we implement the GATv2 attention mechanism as it is provably more powerful and empirically outperforms the GAT attention mechanism. Just like in GAT we employ multiple attention heads as a form of regularisation, which empirically tends to perform better. The multiple attention heads are then aggregated by taking their mean. Figure 3.1 shows the attention-based transport (with a single attention head) more clearly.

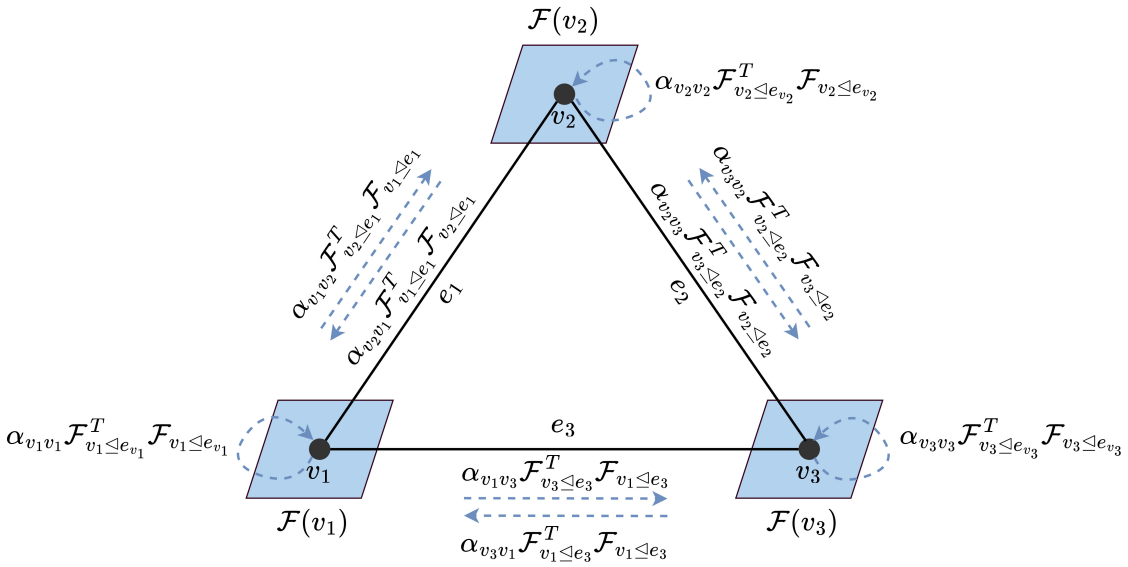


Fig. 3.1 Diagram of the sheaf transport weighted by the attention mechanism (with a single attention head). Transporting a vector from stalk $\mathcal{F}(v_1)$ to $\mathcal{F}(v_2)$ is weighted by $\alpha_{v_1v_2}$, for example. Crucially, the attention weights *from* each node are constrained such that they form a PMF. When the sheaf is trivial, we recover the GAT attention mechanism.

With this new attention machinery over sheaves we have developed, we can now define an attention sheaf diffusion PDE over a graph.

Definition 3.1.1. We define our attention sheaf diffusion equation over graph features:

$$\mathbf{X}(0) = \mathbf{X}, \qquad \frac{\partial}{\partial t}\mathbf{X}(t) = \left(\Lambda\left(\mathbf{X}(t)\right) \otimes \mathbf{1}_d \odot \mathbf{A}_{\mathcal{F}} - \mathbf{I}\right)\mathbf{X}(t) \tag{3.6}$$

where $\mathbf{X} \in \mathbb{R}^{(nd)\times f}$ is the graph's feature matrix, $\mathbf{A}_{\mathcal{F}}$ is the sheaf adjacency matrix, $\Lambda\left(\mathbf{X}(t)\right)$ is the feature-and-time dependent *attention matrix*, $\odot$ denotes element-wise multiplication and $\mathbf{1}_d$ denotes a $d \times d$ matrix where all entries are 1.

It is common to want to analyse the convergence properties of time-dependent PDEs as $t \to \infty$. This particular PDE is complicated to study and for this reason we need to make some simplifications to treat it mathematically. One particular complexity comes from the $\Lambda\left(\mathbf{X}(t)\right)$ term as it is time-dependent. When studying the PDE, we therefore make the simplification that $\Lambda\left(\mathbf{X}(t)\right)$ remains fixed at $\mathbf{X}(0)$ such that $\Lambda\left(\mathbf{X}(t)\right) = \Lambda\left(\mathbf{X}(0)\right) = \Lambda$ (simply denoted by Λ for convenience).

We start by showing that the PDE converges when the underlying sheaf is trivial. We do this by demonstrating that the attentive sheaf diffusion PDE is equivalent to the GRAND model when the sheaf is trivial.

Lemma 3.1.2. Given a trivial sheaf $(G, \mathcal{F})$ with connected graph G (and augmented with self-loops), the linear PDE:

$$\mathbf{X}(0) = \mathbf{X}, \quad \frac{\partial}{\partial t}\mathbf{X}(t) = (\Lambda \otimes \mathbf{1}_d \odot \mathbf{A}_{\mathcal{F}} - \mathbf{I})\,\mathbf{X}(t) \tag{3.7}$$

converges in the time limit $t \to \infty$ to the f-dimensional mean vector $\boldsymbol{\mu}$ of the initial features $\mathbf{X}(0)$.

Proof. As the sheaf is trivial then we can re-write (preserving the initial condition) and using the fact that for a trivial sheaf $\mathbf{A}_{\mathcal{F}} = \mathbf{A}_G$:

$$\begin{aligned}
\frac{\partial}{\partial t}\mathbf{X}(t) &= (\Lambda \otimes \mathbf{1}_d \odot \mathbf{A}_{\mathcal{F}} - \mathbf{I})\,\mathbf{X}(t) && (3.8)\\
&= (\Lambda \odot \mathbf{A}_{\mathcal{F}} - \mathbf{I})\,\mathbf{X}(t) && (3.9)\\
&= -(\mathbf{I} - \Lambda \odot \mathbf{A}_{\mathcal{F}})\,\mathbf{X}(t) && (3.10)\\
&= -(\mathbf{I} - \Lambda \odot \mathbf{A}_G)\,\mathbf{X}(t) && (3.11)
\end{aligned}$$

which is the same form of the GRAND model studied by (Di Giovanni et al., 2022). We can use lemma A.2.2 (see appendix) to complete the proof. □

Effectively, what we have just shown is that when the sheaf is trivial, the PDE converges in the time limit to the mean vector of the node features. This result comes from the fact that with a trivial sheaf the attentive sheaf PDE collapses to the GRAND model. This result is another way to motivate our attention mechanism and to ground our model in previous work. Effectively, our model is a sheaf generalisation of GRAND.

If we restrict the restriction maps to be diagonal matrices, then for stalk width $d > 1$ effectively it would be equivalent to having d independent GRAND models, one for each stalk dimension. Instead, since we work with orthogonal restriction maps, the stalk dimensions are not independent anymore and are able to interact with each-other in non-trivial ways. This hints to the fact that the attentive sheaf PDE is more expressive than the GRAND model.

We now study the convergence of the PDE for higher stalk dimensions $d > 1$.

Theorem 3.1.3. Consider a sheaf $(G, \mathcal{F})$ with connected graph G (and augmented with self-loops), and with all restriction $\mathcal{F}_{v \trianglelefteq e}$ maps as regular matrices. The linear PDE:

$$\mathbf{X}(0) = \mathbf{X}, \quad \frac{\partial}{\partial t}\mathbf{X}(t) = (\Lambda \otimes \mathbf{1}_d \odot \mathbf{A}_{\mathcal{F}} - \mathbf{I})\,\mathbf{X}(t) \tag{3.12}$$

then converges to the f-dimensional mean vector $\boldsymbol{\mu}_d$ for each stalk dimension over the node stalks.

Proof. We call a matrix $\mathbf{A}$ *regular*, when for all i and j there exists an n such that $(\mathbf{A}^n)_{ij}$. If the restriction maps are all regular, then the sheaf adjacency matrix (composed of these restriction maps) must be regular as well.
Consider all the stalks to be d-dimensional, then the $nd \times nd$ sheaf Adjacency matrix is a (positively) weighted graph adjacency matrix of a graph with nd nodes and 1-dimensional stalks. In some sense, the additional stalk width could be thought of as as adding virtual nodes to the graph. Our PDE then becomes the PDE in lemma 3.1.2, meaning that it converges to the mean vector $\boldsymbol{\mu}$ of the initial features. Since the initial features are over d-dimensional stalks now, the PDE converges over each stalk dimension to its corresponding mean vector $\boldsymbol{\mu}_d$. □

To summarise, when the sheaf is trivial, the PDE converges to the mean vector $\boldsymbol{\mu}$ of the initial features $\mathbf{X}(0)$. Instead, when we work with arbitrary regular restriction maps, the PDE converges to the mean vector over each stalk dimension $\boldsymbol{\mu}_d$. While the regularity assumption on the restriction maps may seem a bit restrictive at first, it actually helps to directly compare to the trivial sheaf case as we have converge to the same type of structure: the mean vector of the features.

For the trivial sheaf form (equivalent to GRAND), over-smoothing occurs, as also pointed out by Di Giovanni et al. (2022). Instead, our sheaf model is able to leverage the additional stalk width to maintain the additional information without smoothing the signal. This is an analogous argument to the one presented in Bodnar et al. (2022) where they show that graph heat diffusion is prone to over-smoothing, while sheaf heat diffusion is not under specific conditions.

3.2 Sheaf Attention Networks

Having formalised and studied the attentive sheaf diffusion PDE, we now wish to construct discrete GNN layers out of it. To do this, we consider an Euler discretisation with unit time-step ($\tau = 1$) of equation 3.6:

$$\frac{\mathbf{X}(t+1)-\mathbf{X}(t)}{\tau} = (\Lambda(\mathbf{X}(t)) \otimes \mathbf{1}_d \odot \mathbf{A}_{\mathcal{F}} - \mathbf{I})\,\mathbf{X}(t) \tag{3.13}$$

$$\mathbf{X}(t+1) = \mathbf{X}(t) + (\Lambda(\mathbf{X}(t)) \otimes \mathbf{1}_d \odot \mathbf{A}_{\mathcal{F}} - \mathbf{I})\,\mathbf{X}(t) \tag{3.14}$$

$$\mathbf{X}(t+1) = (\Lambda(\mathbf{X}(t)) \otimes \mathbf{1}_d \odot \mathbf{A}_{\mathcal{F}})\,\mathbf{X}(t) \tag{3.15}$$

We can equip equation 3.15 with weight matrices $\mathbf{W}_1 \in \mathbb{R}^{d\times d}$ and $\mathbf{W}_2 \in \mathbb{R}^{f_{t-1}\times f_t}$ and a non-linearity σ to derive our new *Sheaf Attention Network* (SAN) layer. The weigh matrices are distinct for each layer but we suppress that from the notation for convenience from here onwards.

Definition 3.2.1. The Sheaf Attention Network (SAN) layer takes the form:

$$\mathbf{X}_t = \sigma((\Lambda(\mathbf{X}_{t-1}) \otimes \mathbf{1}_d \odot \mathbf{A}_{\mathcal{F}})(\mathbf{I}_n \otimes \mathbf{W}_1)\,\mathbf{X}_{t-1}\mathbf{W}_2) \tag{3.16}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d\times d}$ and $\mathbf{W}_2 \in \mathbb{R}^{f_{t-1}\times f_t}$ are weight matrices, σ is a non-linear activation function, f_{t-1} and f_t are the input and output channels respectively, and $\Lambda(\mathbf{X}_{t-1})$ is the attention matrix.

Note that by setting σ to an identity mapping and the weight matrices to identity matrices, we recover a direct discrete form of the attentive sheaf diffusion PDE. This suggests that the SAN model is at least as expressive as a direct discretisation, but potentially more given the right choice of σ and weight matrices.

We start by showing that our novel SAN model is a generalisation of the GAT model. In particular, the GAT model is recovered when the sheaf is trivial.

Theorem 3.2.2. The Graph Attention Network (GAT) is a special case of the Sheaf Attention Network (SAN), which arises when the underlying sheaf $(G,\mathcal{F})$ is trivial.

Proof. Recall from lemma 2.3.7 that when the sheaf is trivial, the sheaf Laplacian is the graph Laplacian. When the sheaf is trivial $d=1$, therefore $\mathbf{W}_1 \in \mathbb{R}^{d\times d}$ becomes a scalar (as $d=1$) and $\otimes\mathbf{1}_d$ becomes multiplication by 1, so we can drop both terms. This allows us to write equation 3.16 as:

$$\mathbf{X}_t = \sigma((\Lambda(\mathbf{X}_{t-1}) \otimes \mathbf{1}_d \odot \mathbf{A}_{\mathcal{F}})(\mathbf{I}_n \otimes \mathbf{W}_1)\,\mathbf{X}_{t-1}\mathbf{W}_2) \tag{3.17}$$

$$= \sigma((\Lambda(\mathbf{X}_{t-1}) \odot \mathbf{A}_G)\,\mathbf{X}_{t-1}\mathbf{W}_2) \tag{3.18}$$

The graph adjacency matrix has for entry u, v, 1 if the edge $e = u \to v$ exists and 0 otherwise (including potentially self-loops). The operation $\Lambda(\mathbf{X}_{t-1}) \odot \mathbf{A}_G$ therefore has the effect of making sure that the attention matrix respects the edge structure of the graph - just like in the GAT model. If we take entry u, v to be $\alpha_{u,v}$, then we recover the GAT message passing form:

$$\mathbf{X}_i^{(t)} = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{X}_j^{(t-1)} \mathbf{W}_2\right) \tag{3.19}$$

□

The fact that a trivial sheaf recovers the GAT model from SAN hints to the additional expressiveness of the SAN model. A trivial sheaf, with $d = 1$ and identity restriction maps may be thought of as such when all restriction maps belong to $SO(1)$, the special orthogonal group of dimension 1. This is because the only element in $SO(1)$ is the element 1, imposed from the fact that the determinant must be 1 and the determinant of a 1×1 matrix is the number itself.

Now consider restriction maps part of the orthogonal group $O(1)$ instead of special orthogonal group $SO(1)$. We have that the group elements of $O(1)$ are the set $\{1, -1\}$ instead of the much more restrictive group $SO(1)$ with group element simply $\{1\}$. By allowing for signed restriction maps, we recover the signed GAT model (Huang et al., 2019), which is able to give either positive or negative attention weights to edges.

One can imagine an $O(d)$-SAN model as a generalisation of a signed GAT, where now the restriction maps model higher dimensional rotations, which can be "positive" or "negative" with respect to multiple axes based on the rotation angles.

For these reasons, in all our experiments we make the restriction that all of the restriction maps are elements of the d-dimensional orthogonal group, i.e. $\mathcal{F}_{v \trianglelefteq e} \in O(d)$, effectively learning an $O(d)$-vector bundle.

There are several advantages which come from using strictly orthogonal maps, Bodnar et al. (2022) for example show that orthogonal maps are more efficient with the stalk dimension than general or diagonal linear maps. Another advantage is that effectively an orthogonal linear map, as previously discussed, is $d(d-1)/2$-dimensional instead of d^2-dimensional in the general case. This means that we are in practice learning less parameters which helps with overfitting. Additionally, orthogonal maps, having a determinant of ± 1, make stacking many layers easier to train due to the fact that these maps are not responsible for vanishing or exploding gradients.

Encoding signed messages (using rotation angles of orthogonal maps for example) is crucial in settings of low homophily (Yan et al., 2021). From an opinion dynamics

perspective, a negatively signed message may be thought of as modelling a node's opinion which contradicts another node's opinion. As one can imagine, this behaviour is related to low homophily, where two nodes are more likely nodes to "disagree" on their class. We therefore expect this higher-dimensional disagreement in the form of higher-dimensional orthogonal restriction maps to be very advantageous in heterophilic settings.

3.3 Attentive Neural Sheaf Diffusion

The previous parameterisation, as we showed, can be thought of as a direct generalisation of the GAT model to sheaves. We now introduce another family of neural models which instead more similarly resembles the neural sheaf diffusion model by Bodnar et al. (2022). This is done in an attempt to more directly compare the sheaf attention mechanism structure to the neural sheaf diffusion models.

Consider the following Euler discretisation with unit time-step ($\tau = 1$) of equation 3.6:

$$\frac{\mathbf{X}(t+1) - \mathbf{X}(t)}{\tau} = \left(\Lambda\left(\mathbf{X}(t)\right) \otimes \mathbf{1}_d \odot \mathbf{A}_{\mathcal{F}} - \mathbf{I}\right)\mathbf{X}(t) \tag{3.20}$$

$$\mathbf{X}(t+1) = \mathbf{X}(t) + \left(\Lambda\left(\mathbf{X}(t)\right) \otimes \mathbf{1}_d \odot \mathbf{A}_{\mathcal{F}} - \mathbf{I}\right)\mathbf{X}(t) \tag{3.21}$$

We can equip the right-most term in equation 3.15 with weight matrices $\mathbf{W}_1 \in \mathbb{R}^{d \times d}$ and $\mathbf{W}_2 \in \mathbb{R}^{f_{t-1} \times f_t}$ and a non-linearity σ to derive our new *Attentive Neural Sheaf Diffusion* (ANSD) model. This is similar to the SAN model, but with a more residual parameterisation.

Definition 3.3.1. The Attentive Neural Sheaf Diffusion (ANSD) layer takes the form:

$$\mathbf{X}_t = \mathbf{X}_{t-1} + \sigma\left(\left(\Lambda\left(\mathbf{X}_{t-1}\right) \otimes \mathbf{1}_d \odot \mathbf{A}_{\mathcal{F}} - \mathbf{I}\right)\left(\mathbf{I}_n \otimes \mathbf{W}_1\right)\mathbf{X}_{t-1}\mathbf{W}_2\right) \tag{3.22}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d \times d}$ and $\mathbf{W}_2 \in \mathbb{R}^{f_{t-1} \times f_t}$ are weight matrices, σ is a non-linear activation function, f_{t-1} and f_t are the input and output channels respectively, and $\Lambda\left(\mathbf{X}_{t-1}\right)$ is the attention matrix.

Remark. One training trick which we use (also used by Bodnar et al. (2022)) is to learn a d-dimensional vector $\boldsymbol{\varepsilon} \in [-1,1]^d$ such that:

$$\mathbf{X}_t = (\mathbf{1} + \boldsymbol{\varepsilon})\mathbf{X}_{t-1} + \sigma\left((\Lambda\left(\mathbf{X}_{t-1}\right) \otimes \mathbf{1}_d \odot \mathbf{A}_{\mathcal{F}} - \mathbf{I}\right)\left(\mathbf{I}_n \otimes \mathbf{W}_1\right)\mathbf{X}_{t-1}\mathbf{W}_2\right) \tag{3.23}$$

which allows the model to adjust the relative weighting of each stalk dimension.

The ANSD model is different to the SAN model as it uses a type of residual parameterisation, where the feature updates are of the form $\mathbf{h}_t = \mathbf{h}_t + f(\mathbf{h}_t, \theta_t)$. Empirically, this type of construction outperforms the previously introduced parameterisation of the Sheaf Attention Network. Theoretically, this increased performance has been motivated from the spectral perspective of being able to act as both a high-pass and low-pass filter Di Giovanni et al. (2022). In particular, without a residual connection the model may be dominated by the low frequency signals over the graph as more layers are added, contributing to oversmoothing.

Summary

We have presented an Attentive Sheaf Diffusion PDE and proposed two separate discretisations and parameterisations leading to the Sheaf Attention Network and Attentive Neural Sheaf Diffusion models. The SAN model is a generalisation of GATs to sheaves, while the ANSD model is an attentive cousin of Neural Sheaf Diffusion. In the next part of the chapter we will evaluate these two models in depth, highlighting both strengths and limitations.

3.4 Evaluation

In this section we evaluate extensively the ANSD and SAN models. In our experiments, we focus on the direct comparison of our sheaf attention mechanisms to GAT and the original Neural Sheaf Diffusion model. In particular, we show that the sheaf attention mechanisms greatly outperform GAT, especially when it comes to the issues of over-smoothing and heterophily. On top of this, sheaf attention mechanisms show very competitive results with state-of-the-art models. Our evaluation includes applications to molecules and a discussion of the limitations. Appendix B contains supplementary information useful for the experimental section, including dataset information and hyper-parameter search space details useful to reproduce the results.

Figure 3.2 shows the general pipeline used in the experiments. First the node features are passed through an MLP ϕ which effectively increases the dimensionality

to the stalk width for each feature channel. For example, if the original node features are n-dimensional and the stalk dimension is d, ϕ augments the feature vectors to nd-dimensional vector spaces. Next, we learn the attention coefficients and the $d \times d$ restriction maps. The former are used to construct the attention matrix Λ, while the latter are used to construct the sheaf adjacency matrix $\mathbf{A}_{\mathcal{F}}$. Finally, we use the SAN or ANSD parameterisation to predict the latent features and evaluate our models.

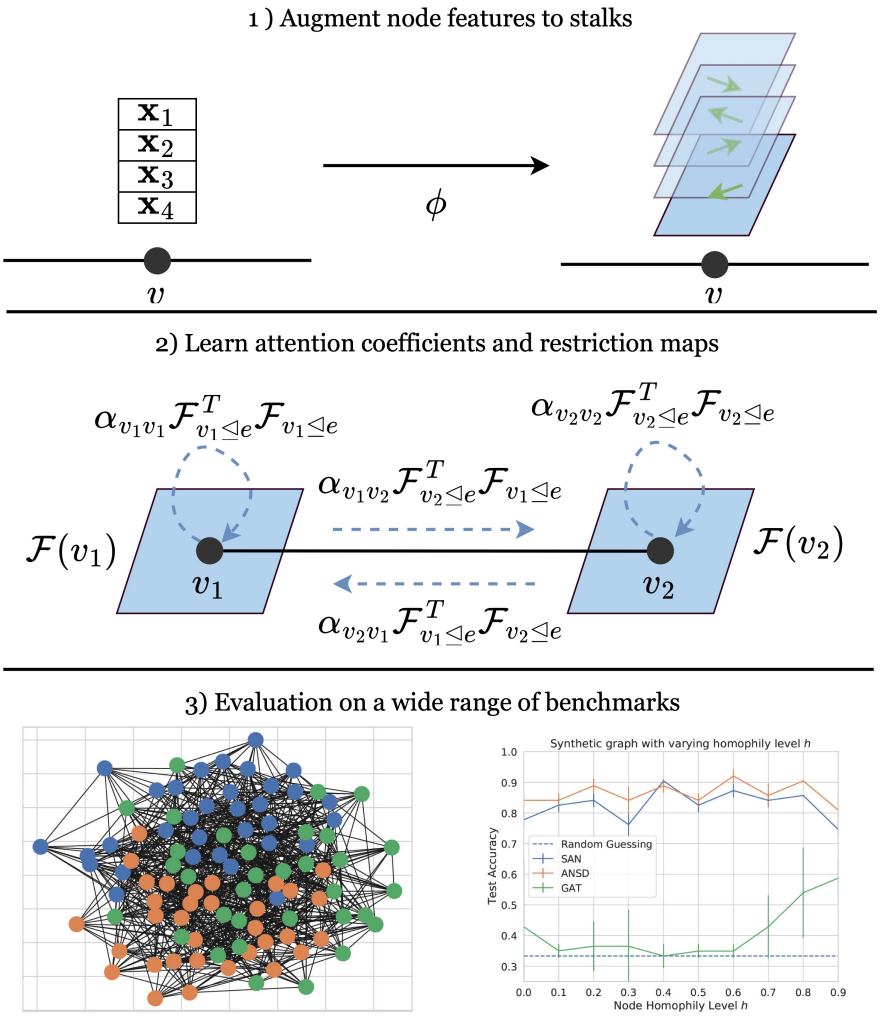


Fig. 3.2 General pipeline used in our experiments. First we map the feature vectors to higher-dimensional stalks via an MLP ϕ. Next, we learn attention coefficients and restriction maps. We then use either the SAN or ANSD parameterisation to make predictions and evaluate our models.

Table 3.1 Accuracy ± stdev for various node classification datasets. The SAN and ANSD accuracies are reported with varying stalk dimension d, ranging from 2 to 16. These accuracies are to be compared with respect to the baseline GAT accuracy. The top three models for each dataset are coloured by **First**, **Second** and **Third**, respectively.

	Texas	**Wisconsin**	**Cornell**	**Citeseer**	**Pubmed**	**Cora**
Homophily level	**0.11**	**0.21**	**0.30**	**0.74**	**0.80**	**0.81**
#Nodes	183	251	183	3,327	18,717	2,708
#Edges	295	466	280	4,676	44,327	5,278
#Classes	5	5	5	7	3	6
GAT	52.16±6.63	49.41±4.09	61.89±5.05	76.55±1.23	87.30±1.10	86.33±0.48
SAN ($d=2$)	84.86±4.86	87.06±4.90	**84.59**±5.55	75.89±1.64	**89.08**±0.37	86.48±1.06
SAN ($d=4$)	**86.22**±4.90	**88.24**±2.77	**84.59**±4.69	76.34±1.56	89.01±0.42	86.84±1.35
SAN ($d=8$)	**85.14**±3.68	87.25±4.32	84.32±6.60	76.33±1.72	**89.12**±0.36	**87.04**±0.91
SAN ($d=16$)	83.51±5.33	**87.84**±4.00	**84.86**±6.42	76.30±1.87	OOM	86.80±1.00
ANSD ($d=2$)	83.24±4.65	86.86±1.97	**85.14**±5.16	76.30±1.88	88.88±0.47	**86.96**±1.49
ANSD ($d=4$)	**85.68**±4.37	**87.84**±5.10	84.32±5.10	**76.58**±1.78	89.00±0.40	**87.10**±1.35
ANSD ($d=8$)	84.05±2.82	**87.65**±4.48	82.97±5.41	**76.79**±1.38	**89.03**±0.42	86.86±1.26
ANSD ($d=16$)	81.35±7.30	87.45±2.35	**85.14**±5.70	**76.64**±1.12	OOM	86.70±0.98

3.4.1 Stalk Width

We start by evaluating the effect increasing the stalk width has on accuracy. Stalk width is a trade-off between higher expressive power and the risk of over-fitting due to the additional dimensionality of the restriction maps. We evaluate on the datasets the GAT model was originally evaluated on: Citeseer, Pubmed and Cora which are all highly homophilic. We additionally also evaluate on Texas, Wisconsin and Cornell to include more heterophilic benchmarks. Further details on these datasets may be found in appendix B.1.1.

As we have shown, the GAT model is a special case of the SAN model, which is recovered with a trivial sheaf. We empirically verify whether the higher stalk dimension is beneficial and results in higher accuracy on the aforementioned datasets.

Table 3.1 shows the results obtained by varying the the stalk dimension. On the highly heterophilic datasets (Texas, Wisconsin and Cornell) SAN and ANSD greatly outperform the GAT model. On the more homophilic datasets, the GAT model becomes more competitive, although the sheaf attention models still seem to be superior.

A stalk width of $d=4$ seems to be a very competitive middle ground and tends to outperform $d=2$. Going beyond $d=4$ is also viable although the performance sometimes is significantly worsened. We attribute this to overfitting due to the higher

number of parameters. On top of this, the datasets we evaluate on (especially the 3 most heterophilic ones) are relatively small which may amplify the overfitting and further hinder the performance of larger stalk widths. Another drawback of a large stalk width is the additional memory costs which may become problematic, as seen with Pubmed and $d = 16$.

The sheaf dimension mechanisms seem to outperform the GAT model overall. The fact that they do so even with a small stalk width d implies that in practice we do not need much additional overhead (introduced by high stalk dimensionality). These results are very much aligned with the work by Bodnar et al. (2022) which shows that $d > 1$ is necessary for higher performance in heterophilic settings.

3.4.2 Oversmoothing

After having found evidence that the additional stalk width indeed does help empirically, we now turn our eye to the oversmoothing issue. We follow the analysis structure done by Yan et al. (2021) by evaluating our sheaf attention models on Cora, Citeseer, Cornell and Chameleon with layers increasing in powers of 2 from 2 to 64.

The GNN models evaluated are divided in three classes:

1. Classical: GCN Kipf and Welling (2016), and GAT Velickovic et al. (2017).
2. Models for heterophilic settings: GGCN Yan et al. (2021), Geom-GCN Pei et al. (2020), H2GCN Zhu et al. (2020), and GPRGNN Chien et al. (2020).
3. Models which address over-smoothing: GCNII Chen et al. (2020a), and PairNorm Zhao and Akoglu (2019).

The models are evaluated from 2 to 64 layers. The results in the table are taken from Yan et al. (2021) and we evaluate our models on the same splits utilised in their evaluation for consistency.

Table 3.2 shows the oversmoothing results. We see how models which are not designed for oversmoothing exhibit various issues, ranging from accuracy dropping rapidly (GCN and Geom-GCN), to memory issues (H2GCN), to numerical instability (GAT).

We find that our sheaf attention mechanism models perform very competitively when compared to state of the art techniques such as GCNII. This is a very exciting result as it shows that sheaf attention mechanisms can go much deeper than traditional ones such as GAT. Not only do they maintain their performance as layers are added, they achieve very high performance in general whilst doing so. We attribute this to

Table 3.2 Accuracy ± stdev for various node classification datasets. Accuracy is reported from 2 to 64 layers increasing in powers of 2. The "best" results corresponds to the number of layers which resulted in the highest accurracy. "OOM" stands for out of memory, whilst "INS" stands for numerical instability. The top three models for each dataset and layer count are coloured by **First**, **Second** and **Third**, respectively.

Layers	2	4	8	16	32	64	Best
	Cora (h=0.81)						
SAN (ours)	86.90±1.31	86.84±0.97	86.68±1.13	86.54±0.89	**86.62**±1.39	86.26±1.09	2
ANSD (ours)	86.98±1.07	**87.08**±1.26	**86.80**±1.15	86.84±1.24	86.56±0.75	86.76±1.49	4
GGCN	87.00±1.15	**87.48**±1.32	**87.63**±1.33	**87.51**±1.19	**87.95**±1.05	**87.28**±1.41	32
GPRGNN	**87.93**±1.11	**87.95**±1.18	**87.87**±1.41	**87.26**±1.51	**87.18**±1.29	**87.32**±1.21	4
H2GCN*	**87.87**±1.20	86.10±1.51	86.18±2.10	OOM	OOM	OOM	2
GCNII*	85.35±1.56	85.35±1.48	86.38±0.98	**87.12**±1.11	**87.95**±1.23	**88.37**±1.25	64
PairNorm	85.79±1.01	85.07±0.91	84.65±1.09	82.21±2.84	60.32±8.28	44.39±5.60	2
Geom-GCN*	85.35±1.57	21.01±2.61	13.98±1.48	13.98±1.48	13.98±1.48	13.98±1.48	2
GCN	86.98±1.27	83.24±1.56	31.03±3.08	31.05±2.36	30.76±3.43	31.89±2.08	2
GAT	**87.30**±1.10	86.50±1.20	84.97±1.24	INS	INS	INS	2
	Citeseer (h=0.74)						
SAN (ours)	76.27±1.76	76.30±1.80	**76.62**±1.70	76.18±1.47	76.07±2.18	75.99±1.84	8
ANSD (ours)	**76.99**±1.74	**76.86**±1.71	76.61±1.51	**76.69**±1.56	**76.22**±1.47	**76.44**±1.30	2
GGCN	76.83±1.82	**76.77**±1.48	**76.91**±1.56	**76.88**±1.56	**76.97**±1.52	**76.65**±1.38	10
GPRGNN	**77.13**±1.67	**77.05**±1.43	**77.09**±1.62	76.00±1.64	74.97±1.47	74.41±1.65	2
H2GCN*	76.90±1.80	76.09±1.54	74.10±1.83	OOM	OOM	OOM	1
GCNII*	75.42±1.78	75.29±1.90	76.00±1.66	**76.96**±1.38	**77.33**±1.48	**77.18**±1.47	32
PairNorm	73.59±1.47	72.62±1.97	72.32±1.58	59.71±15.97	27.21±10.95	23.82±6.64	2
Geom-GCN*	**78.02**±1.15	23.01±1.95	7.23±0.87	7.23±0.87	7.23±0.87	7.23±0.87	2
GCN	76.50±1.36	64.33±8.27	24.18±1.71	23.07±2.95	25.3±1.77	24.73±1.66	2
GAT	76.55±1.23	75.33±1.39	66.57±5.08	INS	INS	INS	2
	Cornell (h=0.3)						
SAN (ours)	**82.70**±6.64	**84.59**±4.69	**85.68**±4.53	**84.32**±6.82	**83.51**±7.20	**78.65**±10.29	8
ANSD (ours)	**84.86**±6.07	**84.32**±5.10	**84.86**±5.95	**84.59**±6.51	**83.24**±3.97	**82.43**±8.90	8
GGCN	**83.78**±6.73	**83.78**±6.16	**84.86**±5.69	**83.78**±6.73	**83.78**±6.51	**84.32**±5.90	6
GPRGNN	76.76±8.22	77.57±7.46	**80.27**±8.11	78.38±6.04	74.59±7.66	70.00±5.73	8
H2GCN*	81.89±5.98	82.70±6.27	**80.27**±6.63	OOM	OOM	OOM	1
GCNII*	67.57±11.34	64.59±9.63	73.24±5.91	77.84±3.97	75.41±5.47	73.78±4.37	16
PairNorm	50.27±7.17	53.51±8.00	58.38±5.01	58.38±3.01	58.92±3.15	58.92±3.15	32
Geom-GCN*	60.54±3.67	23.78±11.64	12.97±2.91	12.97±2.91	12.97±2.91	12.97±2.91	2
GCN	60.54±5.30	59.19±3.30	58.92±3.15	58.92±3.15	58.92±3.15	58.92±3.15	2
GAT	61.89±5.05	58.38±4.05	58.38±3.86	INS	INS	INS	2
	Chameleon (h=0.23)						
SAN (ours)	**65.88**±2.10	**65.99**±1.28	**68.16**±2.18	**68.62**±2.81	**67.61**±2.80	OOM	16
ANSD (ours)	**65.35**±1.26	**67.11**±1.88	**66.69**±2.30	**67.39**±1.84	**66.91**±1.61	OOM	16
GGCN	**70.77**±1.42	**69.58**±2.68	**70.33**±1.70	**70.44**±1.82	**70.29**±1.62	**70.20**±1.95	5
GPRGNN	46.58±1.771	45.72±3.45	41.16±5.79	39.58±7.85	35.42±8.52	36.38±2.40	2
H2GCN*	59.06±1.85	60.11±2.15	OOM	OOM	OOM	OOM	4
GCNII*	61.07±4.10	63.86±3.04	62.89±1.18	60.20±2.10	56.97±1.81	**55.99**±2.27	4
PairNorm	62.74±2.82	59.01±2.80	54.12±2.24	46.38±2.23	46.78±2.26	**46.27**±3.24	2
Geom-GCN*	60.00±2.81	19.17±1.66	19.58±1.73	19.58±1.73	19.58±1.73	19.58±1.73	2
GCN	64.82±2.24	53.11±4.44	35.15±3.14	35.39±3.23	35.20±3.25	35.50±3.08	2
GAT	60.26±2.50	48.71±2.96	35.09±3.55	INS	INS	INS	2

the more complex geometry of the underlying sheaf. Our sheaf models leverage the additional stalk width to mitigate the smoothing of the signal.

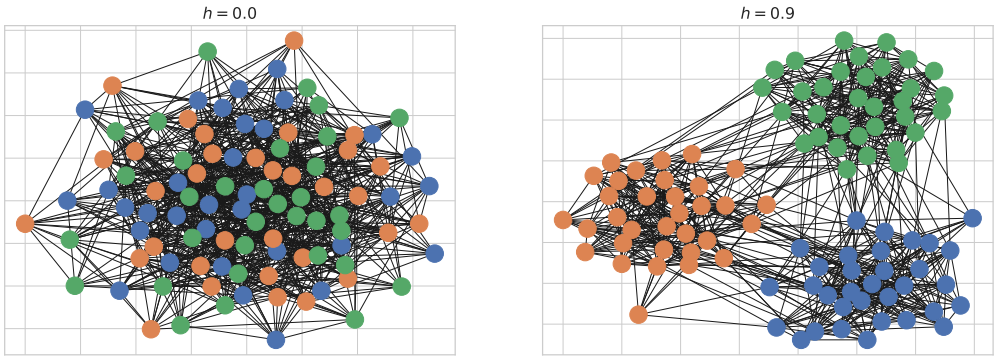
One very satisfying trend is that our attentive sheaf models perform extremely well on the Cornell and Chameleon datasets, which are particularly heterophilic. This suggests that while the other models are designed either for oversmoothing or for heterophily, our models are particularly well suited for both tasks at once.

3.4.3 Heterophily

Heterophily: Synthetic Benchmark

To compare in a controlled environment the perfomance between GAT and the sheaf attention mechanisms in heterophilic settings, we devise a synthetic benchmark. We generate graphs with 3 classes and sample edges with a progressively higher homophilic probability h. The feature vectors are 3-dimensional and each dimension is sampled from an independent Gaussian distribution.

Figure 3.3 shows the results of this process for the most heterophilic ($h = 0.0$) and the most homophilic graphs generated ($h = 0.9$). The full spectrum of synthetic graphs, alongside further generation details, may be found in appendix B.2.



(a) Least heterophilic synthetic graph with node homophily $h = 0.0$.

(b) Most heterophilic synthetic graph with node homophily $h = 0.9$.

Fig. 3.3 Example of the two edge-case synthetic graphs generated, one with the lowest homophily level $h = 0.0$ (a) and the other with the highest homophily level $h = 0.9$ (b). High homophily results in a graph which has very clearly distinguishable classes, whilst low homophily results in a much more visually noisy graph in comparison.

The goal of this synthetic benchmark is to compare the separation power between GAT and the higher dimensional sheaf attention mechanisms in very controlled conditions of homophily. Figure 3.4 shows the results on the synthetic benchmark.

The GAT model seems to significantly struggle with $h < 0.7$, essentially random guessing with accuracy close to $\frac{1}{3}$. It is only in very high homophily settings that the GAT model is able to perform better than random guessing. The SAN and ANSD models instead have high performance independently of the homophily level. The variance also remains relatively low, suggesting that they are less prone to initialisation issues.

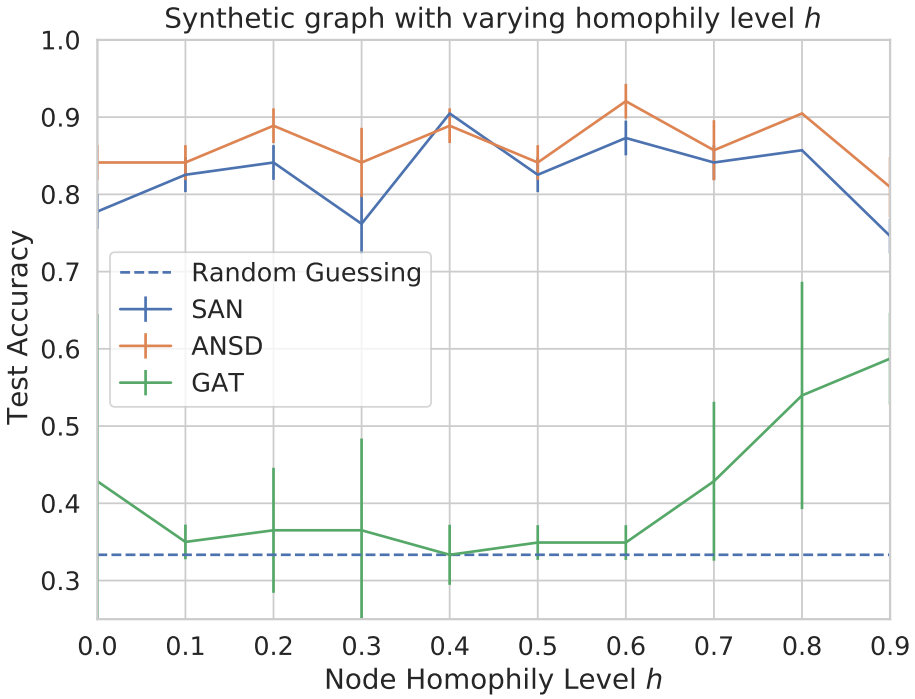


Fig. 3.4 Test accuracy on our synthetic benchmark of 3 classes. Accuracy of $\frac{1}{3}$ is random guessing. The accuracy is plotted as a function of the node homophily level h, which ranges from 0.0 to 0.9 with a step size of 0.1.

We attribute the increased performance to the higher stalk dimension d and the fact that the sheaf models are able to learn "negative" relationships by adjusting the rotation angles. This becomes crucial in heterophilic settings and this is made clear by the synthetic benchmark.

Heterophily: Real-World Datasets

We now evaluate the sheaf attention models on a wide range of real-world datasets, with varying levels of node homophily h and compare the performance to a multitude of different models. In doing so, we remain consistent by using the same benchmarks used by Bodnar et al. (2022) to evaluate the original NSD model. The datasets are

real-world datasets which aim to evaluate the performance of models in heterophilic settings (Pei et al., 2020; Rozemberczki et al., 2021). Further details on these datasets may be found in appendix B.1.1.

The datasets are ordered based on their node homophily coefficient $0 \leq h \leq 1$, which is higher for more homophilic datasets. The quantity h is the fraction of edges which connect nodes of the same class label, known as the edge homophily level.

The results are collected over 10 fixed splits, where 48%, 32%, and 20% of nodes per class are used for training, validation, and testing, respectively. The reported score is the test set accuracy, which corresponds to the tuned model with the highest validation accuracy, to avoid data snooping. The details on how the hyper parameters were searched may be found in appendix B.3.

Table 3.3 shows the test accuracy results for a wide range of models for the node classification tasks over 9 data-sets, ordered by homophily level. An important baseline is the Multi-Layer Perceptron (MLP), whose result is shown on the last row of table 3.3. The MLP has access only to the node features and it provides an idea of how much useful information GNNs can extract from the graph structure.

The GNN models in table 3.3 may be considered under 3 classes:

1. Classical: GCN Kipf and Welling (2016), GAT Velickovic et al. (2017), and GraphSAGE Hamilton et al. (2017).
2. Models for heterophilic settings: GGCN Yan et al. (2021), Geom-GCN Pei et al. (2020), H2GCN Zhu et al. (2020), GPRGNN Chien et al. (2020), FAGCN Bo et al. (2021), and MixHop Abu-El-Haija et al. (2019).
3. Models which address over-smoothing: GCNII Chen et al. (2020a), and PairNorm Zhao and Akoglu (2019).

Additionally, also the results from Bodnar et al. (2022) with neural sheaf diffusion are included as NSD. The three variants denote different types of restriction maps, namely diagonal (Diag-NSD), orthogonal ($O(d)$-NSD) and general (Gen-NSD). Another direct comparison of interest is the GAT model. The sheaf attention models tie concepts from the NSD and GAT models together, incorporating the cellular sheaf construction from the former and the attention mechanism for the other. The direct comparison between these two aforementioned model is therefore crucial.

The results are reported in table 3.3. As expected, the GAT model performs very poorly on datasets with higher levels of heterophily. For example, in the Texas and Wisconsin datasets the GAT model is significantly outperformed even by a simple MLP. It is clear that the GAT model has a strong inductive bias for homophily and therefore performs very poorly when working in heterophilic settings.

Table 3.3 Accuracy ± stdev for various node classification datasets and models. The datasets are sorted by increasing order of homophily. Our technique is denoted Conn-NSD, while the other Sheaf Diffusion models are Diag-NSD, O(d)-NSD and Gen-NSD. The top three models for each dataset are coloured by **First**, **Second** and **Third**, respectively. The first section includes sheaf-based models, while the second includes other GNN models.

	Texas	**Wisconsin**	**Film**	**Squirrel**	**Chameleon**	**Cornell**	**Citeseer**	**Pubmed**	**Cora**
Homophily level	**0.11**	**0.21**	**0.22**	**0.22**	**0.23**	**0.30**	**0.74**	**0.80**	**0.81**
#Nodes	183	251	7,600	5,201	2,277	183	3,327	18,717	2,708
#Edges	295	466	26,752	198,493	31,421	280	4,676	44,327	5,278
#Classes	5	5	5	5	5	5	7	3	6
SAN (ours)	84.05±5.33	86.47±3.87	37.09±1.18	50.96±1.40	67.46±1.90	84.32±5.64	72.57±1.50	87.12±0.30	85.90±1.85
ANSD (ours)	**85.68**±4.69	87.45±3.19	37.66±1.40	54.39±1.76	**68.38**±2.14	84.59±5.93	76.81±1.82	89.21±0.37	87.20±1.03
Diag-NSD	**85.67**±6.95	**88.63**±2.75	**37.79**±1.01	**54.78**±1.81	**68.68**±1.73	**86.49**±7.35	**77.14**±1.85	89.42±0.43	87.14±1.06
$O(d)$-NSD	**85.95**±5.51	**89.41**±4.74	**37.81**±1.15	**56.34**±1.32	68.04±1.58	**84.86**±4.71	76.70±1.57	**89.49**±0.40	86.90±1.13
Gen-NSD	82.97±5.13	**89.21**±3.84	**37.80**±1.22	53.17±1.31	67.93±1.58	**85.68**±6.51	76.32±1.65	89.33±0.35	87.30±1.15
GGCN	84.86±4.55	86.86±3.29	37.54±1.56	**55.17**±1.58	**71.14**±1.84	**85.68**±6.63	**77.14**±1.45	89.15±0.37	**87.95**±1.05
H2GCN	84.86±7.23	87.65±4.98	35.70±1.00	36.48±1.86	60.11±2.15	82.70±5.28	77.11±1.57	**89.49**±0.38	**87.87**±1.20
GPRGNN	78.38±4.36	82.94±4.21	34.63±1.22	31.61±1.24	46.58±1.71	80.27±8.11	77.13±1.67	87.54±0.38	**87.95**±1.18
FAGCN	82.43±6.89	82.94±7.95	34.87±1.25	42.59±0.79	55.22±3.19	79.19±9.79	N/A	N/A	N/A
MixHop	77.84±7.73	75.88±4.90	32.22±2.34	43.80±1.48	60.50±2.53	73.51±6.34	76.26±1.33	85.31±0.61	87.61±0.85
GCNII	77.57±3.83	80.39±3.40	37.44±1.30	38.47±1.58	63.86±3.04	77.86±3.79	**77.33**±1.48	**90.15**±0.43	**88.37**±1.25
Geom-GCN	66.76±2.72	64.51±3.66	31.59±1.15	38.15±0.92	60.00±2.81	60.54±3.67	**78.02**±1.15	**89.95**±0.47	85.35±1.57
PairNorm	60.27±4.34	48.43±6.14	27.40±1.24	50.44±2.04	62.74±2.82	58.92±3.15	73.59±1.47	87.53±0.44	85.79±1.01
GraphSAGE	82.43±6.14	81.18±5.56	34.23±0.99	41.61±0.74	58.73±1.68	75.95±5.01	76.04±1.30	88.45±0.50	86.90±1.04
GCN	55.14±5.16	51.76±3.06	27.32±1.10	53.43±2.01	64.82±2.24	60.54±5.30	76.50±1.36	88.42±0.50	86.98±1.27
GAT	52.16±6.63	49.41±4.09	27.44±0.89	40.72±1.55	60.26±2.50	61.89±5.05	76.55±1.23	87.30±1.10	86.33±0.48
MLP	80.81±4.75	85.29±3.31	36.53±0.70	28.77±1.56	46.21±2.99	81.89±6.40	74.02±1.90	75.69±2.00	87.16±0.37

Instead, the sheaf attention models seem to perform very competitively across the board and especially in heterophilic settings when compared to a wide variety of state-of-the-art models. This gives further empirical evidence that the attention mechanisms, which now act on higher-dimensional non-trivial sheaves, are indeed useful. In particular, the ANSD model outperforms the GAT on every dataset, with larger empirical gains as homophily decreases. This result is consistent with the theory developed in Bodnar et al. (2022), which suggests that higher dimensional stalks are needed in heterophilic settings.

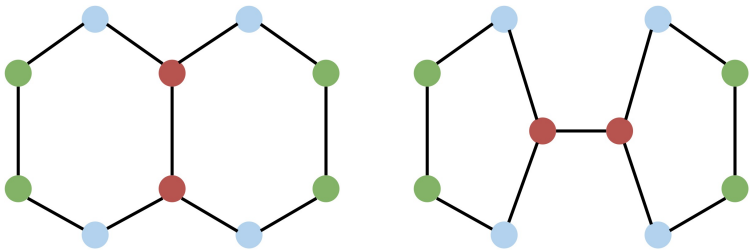
We also find that the ANSD model tends to outperform the SAN parameterisation. This may be explained from the theoretical analysis by Di Giovanni et al. (2022), which suggests that residual parameterisations are better equipped for high frequency signals over graphs. Overall, these results show how effective the sheaf attention mechanisms are, achieving competitive performance when compared to state-of-the-art models on a wide range of domains.

3.4.4 Molecular Applications

We started the section by proposing a new attention mechanism based on sheaves and used it to define an attention-based sheaf diffusion equation. We then took two distinct discretisated parameterisations of this process resulting in Sheaf Attention Networks (SAN) and Attentive Neural Sheaf Diffusion (ANSD). We evaluated the SAN and ANSD models on various datasets and synthetic benchmarks to show their empirical strength. In particular, we showed their effectiveness in tackling issues related to oversmoothing and heterophily.

In this section we will discuss one potentially useful real-world application of attention mechanisms over sheaves: molecules. Molecular interactions are extremely complicated to model. Traditional GNN models struggle due to a variety of factors. For example, ring structures are very common in chemical compounds and standard GNN models are ill-equipped to deal with them. Intuitively, message passing is too local to identify rings correctly and would require very deep GNNs. Unfortunately, we already discussed how very deep GNNs leads to the phenomenon of oversmoothing.

Example. One example of a pair of problematic molecules is Decalin and Bicyclopentil. Figure 3.5 shows the two molecular graphs side by side. The two molecular graphs are isomorphic according to the Weisfeiler-Lehman graph isomorphism test (WL test), a procedure which traditional message passing GNNs have been shown to be as expressive as. The two graphs however are clearly not isomorphic, for instance Decalin has two rings of length 6, while Bicyclopentil has two rings of length 5.



(a) Decalin (b) Bicyclopentil

Fig. 3.5 Graphs of Decalin and Bicyclopentil molecules, where nodes are carbon atoms and edges are chemical bonds. The node colours show the stable colouring reached by the Weisfeiler-Lehman algorithm. The two molecular graphs are erroneously considered isomorphic by the WL test, although they are not isomorphic.

These two molecules highlight one of the issues which comes with standard GNNs: models struggle to leverage larger graph sub-structures such as rings due to the local nature of message passing. This is problematic in domains like bio-chemistry as ring

structures are extremely prevalent. There have been many topological approaches which target this issue such as Message Passing Simplicial Networks (Bodnar et al., 2021b) (over Simplicial Complexes) and CW Networks (Bodnar et al., 2021a) (over regular Cell Complexes).

Remark. It is worth mentioning that while traditional message passing GNNs struggle to count substructures both theoretically and empirically (Chen et al., 2020b), there are relatively simple schemes which address some of these issues as well. For instance in our chemical example, one may augment the node feature vectors with ring size information as a pre-processing step. This would allow the GNN to use that information to distinguish the two molecular graphs. Of course the challenge then becomes finding the right quantities to augment the feature vectors with.

As our molecular benchmark, we evaluate on the ZINC dataset. ZINC is a large dataset of molecular graphs, the goal is for each grass to regress a molecular property known as constrained solubility. We follow the experimental setup of suggested by Dwivedi et al. (2020) in which they establish a general methodology to evaluate graph neural networks on various datasets, including ZINC. We use a standard smaller version of ZINC which contains $10,000$ molecules in the training set, $1,000$ in the validation set and $1,000$ in the test set.

The regression task on ZINC, unlike the other datasets, is an inductive problem. Our technique however produces latent feature vectors at each node and not a single output and we must in some way condense all of the node latent vectors into a single regression output. We again follow the standard procedure suggested by Dwivedi et al. (2020), where we compute the mean vector over all the latent nodes and then pass that through an MLP to retrieve a single numerical value (in this case the predicted constrained solubility of the molecule).

Table 3.4 shows the results on the ZINC datasets for the various models evaluated. The models are roughly divided into the following categories:

1. Classical: GCN (Kipf and Welling, 2016), GAT (Velickovic et al., 2017), and GIN (Xu et al., 2018).
2. Residual: GatedGCN (Bresson and Laurent, 2017).
3. Expressivity: PNA (Corso et al., 2020) (aggregation), and DGN (Beaini et al., 2021) (Laplacian eigenvectors).
4. Topological: CIN (Bodnar et al., 2021a), and GSN (Bouritsas et al., 2022).

We find that our technique surpassed the performance of GCNs and GATs. Having said this, it falls short of many of the other techniques. The common denominator

Table 3.4 Mean Average Error (MAE) ± stdev on the ZINC regression task for various models.

Model	ZINC MAE
GCN	0.469±0.002
GAT	0.463±0.002
GatedGCN	0.422±0.006
GIN	0.408±0.008
PNA	0.320±0.032
DGN	0.219±0.010
GSN	0.139±0.007
CIN	0.115±0.003
SAN (ours)	0.434±0.009
ANSD (ours)	0.415±0.011

between the better performing techniques is that they in some way better-equipped to deal with graph substructures such as cycles. For instance, CIN works by considering higher-dimensional (regular) Cell Complexes, topological objects which Simplicial Complexes and graphs are a special case of (see appendix A.1). GSN is also a topologically inspired technique which can detect substructures in graphs.

This is instead not true for our sheaf models. Recall that the type of sheaf we work with is a *cellular* sheaf meaning that this structure is well defined for cell complexes. In our work, we simply utilise the underlying graph, which is an example of a 1-dimensional cell complex. Instead the CWN technique, for example, is equipped for higher dimensional cellular structures. For this reason, we suspect that working over higher dimensional cellular complexes would greatly benefit the sheaf diffusion model. This would of course come at a trade off of the increased computational costs which come from considering the higher dimensional cells.

3.4.5 Limitations

Limitations: Computational Cost We have shown that the sheaf attention models consistently outperform GAT and also greatly help with the oversmoothing and heterophily problems. In particular, they achieve competitive oversmoothing and heterophilic accuracy to state of the art models specifically designed for these tasks. While this is an exciting achievement, there are some drawbacks involved with the additional sheaf structure.

In particular, the models are effectively a higher-dimensional version of GAT. They therefore inherit the costly attention mechanism. A GAT layer has cost $\mathcal{O}(|V|F\hat{F} + |E|\hat{F})$ (Velickovic et al., 2017) where $|V|$ and $|E|$ denote number of nodes and edges respectively, and F and $\hat{F}$ denote the number of input and output features respectively.

With orthogonal restriction maps, we now have an extra $\mathcal{O}(d^3)$ cost due to matrix multiplication with the $d \times d$ restriction maps, similarly to the original NSD work by Bodnar et al. (2022), resulting in $\mathcal{O}(d^3(|V|F\hat{F} + |E|\hat{F}))$. In practice the stalk dimension d is chosen to be small such that $d \leq 5$. Learning $d \times d$ (orthogonal) restriction maps is done efficiently with the Torch Householder library and the matrix operations all support sparse batched GPU computations to alleviate the extra computational complexity.

The memory cost is also significant especially as d grows, as one is required to store (for each of the $|E|$ edges) $d \times d$ restriction maps, which is linear in the number of edges and up to quadratic in d (depending on how the orthogonal matrices are being stored). There is additionally a memory cost from the attention mechanism which scales linearly with number of edges to store the attention coefficients. On top of this, the utilisation of K attention heads multiplies the computational costs of the attention mechanism, although the memory costs remain similar as only the mean of the attention head is required to be store.

The attention sheaf models as a consequence are computationally more expensive than both GAT and the previously propose neural sheaf diffusion model from Bodnar et al. (2022).

Table 3.5 shows the runtime results for the various models being bencharked. These were obtained on 3 synthetic graphs of different sizes. All models only have 1 layer and the sheaf models all have a stalk width $d = 3$. The reported value is the training time in seconds over 100 epochs obtained whilst training on 80% of the samples. The models were trained on an NVIDIA TITAN X GPU and an Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz.

The GAT model is fastest, followed by the Conn-NSD sheaf model (Barbero et al., 2022). The $O(d)$-NSD model comes next, followed by SAN and finally ANSD. The SAN parameterisation is faster than ANSD as it requires fewer operations. On top of this, the neural sheaf diffusion models learn, as previously mentioned, a vector $\boldsymbol{\epsilon} \in [-1, 1]^d$ which adjusts the relative weights of each stalk dimension. Although effective in practice, this adds slight costs when backpropagating.

Overall, we have shown that although the sheaf attention mechanism is slower than the GAT model, the training time is still reasonable especially when compared to the

Table 3.5 Training time (in seconds) for 100 epochs on synthetic graphs of different sizes (lower is better). The models are ordered by speed (faster models on the bottom). $O(d)$-NSD is neural sheaf diffusion with orthogonal restriction maps (Bodnar et al., 2022), whilst Conn-NSD is neural sheaf diffusion with pre-computed connection Laplacians (Barbero et al., 2022). For all the models we utilise 1 layer and stalk width $d = 3$ (when applicable).

	Synthetic 1	**Synthetic 2**	**Synthetic 3**
Rank			
#Nodes	99	999	9,999
#Edges	500	5,000	50,000
ANSD (ours)	0.557	2.620	22.262
SAN (ours)	0.430	2.201	19.544
$O(d)$-NSD	0.417	2.094	17.382
Conn-NSD	0.180	0.839	9.225
GAT	0.128	0.495	3.910

performance gained in practice. The additional attention mechanism overhead from $O(d)$-NSD to ANSD and SAN is also not disastrous.

Limitations: Implementation Complexity Another more subtle limitation is the implementation complexity. For example, constructing the sheaf Laplacian requires the careful juggling of restriction maps which need to be efficiently constrained to be orthogonal. Importantly, the sheaf Laplacian needs to be treated as a sparse object or the technique would run quickly into memory issues. Of course, the additional attention mechanism further complicates this pipeline.

All of this adds extra difficulty when one needs to, for instance, work with batches of graphs in inductive settings. Keeping track of the individual sheaf Laplacians and their corresponding restriction maps becomes much more challenging when compared to more simple GNN methods. Simplifying the implementation complexity of SNNs through sheaf-compatible machine learning libraries is therefore important to allow these models to become more widespread.

Summary

We started the chapter by developing an attention mechanism over sheaves, leading to an attentive sheaf diffusion PDE. We then considered two different parameterisations of the discretised PDE leading to Sheaf Attention Networks and Attentive Neural Sheaf Diffusion. We related all of this to existing GNN literature and showed interesting properties of these constructions. The evaluation section showed that these models address the oversmooothing and heterophily issues present in GAT to a very significant extent. Furthermore, the new models perform very well when compared to current state-of-the-art GNN architectures. We concluded the chapter with a discussion on chemical applications and some limitations of our technique.

CHAPTER 4

CONCLUSION

Yet now I cannot sing out loud,
Peace is my farewell music;
Even crickets are now silent for me,
For Cambridge this evening is silent.

Quietly I am leaving,
Just as quietly as I came;
Gently waving my sleeve,
I am not taking away a single cloud.

Xu Zhimo (1897–1931)

We have reached the end of our topological journey. It is now time to take a step back to recapitulate and reflect upon our findings. In chapter 2, we began with a guided overview of the GNN literature and followed it with a dive into cellular sheaf theory. We then showed how these two, at a glance, very distinct fields can be elegantly tied into one with the concept of sheaf neural networks and neural sheaf diffusion.

In chapter 3, we formalised an attention mechanism over cellular sheaves and used it to construct an attention-based sheaf diffusion PDE. We then studied and evaluated two separate types of discretisations of this equation, one leading to Sheaf Attention Networks and the other leading to Attentive Neural Sheaf Diffusion. We showed how our constructions are generalisations of popular GNN models to sheaves. Importantly, our evaluation supported our theoretical analysis demonstrating the exciting performance SAN and ANSD displayed on oversmoothing and heterophily benchmarks.

In terms of applications, we believe our technique to be especially useful in tasks which are highly heterophilic, such as intruder detection graphs for example. As our analysis showed, the SAN and ANSD models performed remarkably well and significantly improved over the GAT baseline. We also empirically showed that sheaf attention helps address the oversmoothing problem present in GAT. We then explored applications to molecules and believe that applying sheaf neural networks to molecular tasks to be a potentially very productive avenue of research, with the right improvements.

The analysis was structured in such a way to ground our proposal in existing literature. For example, we showed how our models related to the GAT and GRAND models. In doing so, we believe that our SAN and ANSD models are capable of having measurable impact in tasks where these models are already being used today. For example, a SAN model functions as a direct replacement of a GAT, with overall improved performance, at the cost of computational overhead.

4.1 Future Work

We believe to have identified exciting areas of future work, some of which were touched upon in chapter 3. One direction aims to address the computational overhead which comes with learning the sheaf with gradient-based optimisation. Similarly to what was done by Barbero et al. (2022), we believe that pre-computing the orthogonal restriction maps, from a Riemannian geometry perspective, may help with the additional computational costs introduced by the sheaf attention mechanism.

In terms of applications, we believe that sheaf attention mechanisms may become incredibly useful in bio-chemical applications. Molecular interactions are very complex to model and the higher-dimensional sheaf structure may be a great candidate. In our work we focused on a sheaves over 1-dimensional cellular complexes, but we believe that exploring higher-dimensional complexes may be very fruitful to build a sheaf model which is able to take into account cycles and other types of substructures (see appendix A.1 for further details).

Due to the interesting connection between GATs and Transformers, an exciting potential application could be natural language processing. The higher dimensional sheaf structure may prove useful to model complicated interactions between words. Effectively, our models are sheaf generalisations of GAT, which in turn is a graph generalisation of a Transformer. As a consequence, exploring sheaf Transformer models may prove to be very fruitful in a lot of Transformer based tasks.

REFERENCES

Abu-El-Haija, S., Perozzi, B., Kapoor, A., Alipourfard, N., Lerman, K., Harutyunyan, H., Ver Steeg, G., and Galstyan, A. (2019). Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *international conference on machine learning*, pages 21–29. PMLR.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Barbero, F., Bodnar, C., Borde, H. S. d. O., Bronstein, M., Veličković, P., and Liò, P. (2022). Sheaf neural networks with connection laplacians.

Beaini, D., Passaro, S., Létourneau, V., Hamilton, W., Corso, G., and Liò, P. (2021). Directional graph networks. In *International Conference on Machine Learning*, pages 748–758. PMLR.

Bo, D., Wang, X., Shi, C., and Shen, H. (2021). Beyond low-frequency information in graph convolutional networks. *arXiv preprint arXiv:2101.00797*.

Bodnar, C., Di Giovanni, F., Chamberlain, B. P., Lio, P., and Bronstein, M. M. (2022). Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in gnns. In *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*.

Bodnar, C., Frasca, F., Otter, N., Wang, Y., Liò, P., Montufar, G. F., and Bronstein, M. (2021a). Weisfeiler and lehman go cellular: Cw networks. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, pages 2625–2640. Curran Associates, Inc.

Bodnar, C., Frasca, F., Wang, Y., Otter, N., Montufar, G. F., Lió, P., and Bronstein, M. (2021b). Weisfeiler and lehman go topological: Message passing simplicial networks. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 1026–1037. PMLR.

Bouritsas, G., Frasca, F., Zafeiriou, S. P., and Bronstein, M. (2022). Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Bresson, X. and Laurent, T. (2017). Residual gated graph convnets. *arXiv preprint arXiv:1711.07553*.

Brody, S., Alon, U., and Yahav, E. (2021). How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*.

Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. (2021). Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*.

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

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. (2013). Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*.

Chamberlain, B., Rowbottom, J., Gorinova, M. I., Bronstein, M., Webb, S., and Rossi, E. (2021). Grand: Graph neural diffusion. In *International Conference on Machine Learning*, pages 1407–1418. PMLR.

Chen, M., Wei, Z., Huang, Z., Ding, B., and Li, Y. (2020a). Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, pages 1725–1735. PMLR.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural ordinary differential equations. *Advances in neural information processing systems*, 31.

Chen, Z., Chen, L., Villar, S., and Bruna, J. (2020b). Can graph neural networks count substructures? *Advances in neural information processing systems*, 33:10383–10395.

Cheng, J., Dong, L., and Lapata, M. (2016). Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.

Chien, E., Peng, J., Li, P., and Milenkovic, O. (2020). Joint adaptive feature smoothing and topology extraction via generalized pagerank gnns. *arXiv preprint arXiv:2006.07988*.

Corso, G., Cavalleri, L., Beaini, D., Liò, P., and Veličković, P. (2020). Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems*, 33:13260–13271.

Di Giovanni, F., Rowbottom, J., Chamberlain, B. P., Markovich, T., and Bronstein, M. M. (2022). Graph neural networks as gradient flows. *arXiv preprint arXiv:2206.10991*.

Dwivedi, V. P., Joshi, C. K., Laurent, T., Bengio, Y., and Bresson, X. (2020). Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*.

Hamermesh, M. (2005). Group theory. *Mathematical Tools for Physicists*, pages 189–212.

Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

Hansen, J. (2020). *Laplacians of Cellular Sheaves: Theory and Applications*. PhD thesis, University of Pennsylvania.

Hansen, J. and Gebhart, T. (2020). Sheaf neural networks. *arXiv preprint arXiv:2012.06333*.

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

Hansen, J. and Ghrist, R. (2021). Opinion dynamics on discourse sheaves. *SIAM Journal on Applied Mathematics*, 81(5):2033–2060.

Hatcher, A. (2005). *Algebraic topology*. .

Henaff, M., Bruna, J., and LeCun, Y. (2015). Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*.

Huang, J., Shen, H., Hou, L., and Cheng, X. (2019). Signed graph attention networks. In *International Conference on Artificial Neural Networks*, pages 566–577. Springer.

Joshi, C. (2020). Transformers are graph neural networks. *The Gradient*, page 5.

Kelley, J. L. (2017). *General topology*. Courier Dover Publications.

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

Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Lin, Z., Feng, M., Santos, C. N. d., Yu, M., Xiang, B., Zhou, B., and Bengio, Y. (2017). A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Pandit, S., Chau, D. H., Wang, S., and Faloutsos, C. (2007). Netprobe: a fast and scalable system for fraud detection in online auction networks. In *Proceedings of the 16th international conference on World Wide Web*, pages 201–210.

Pei, H., Wei, B., Chang, K. C.-C., Lei, Y., and Yang, B. (2020). Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*.

Rozemberczki, B., Allen, C., and Sarkar, R. (2021). Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2008). The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80.

Singer, A. and Wu, H.-T. (2012). Vector diffusion maps and the connection laplacian. *Communications on pure and applied mathematics*, 65(8):1067–1144.

Tu, L. W. (2011). Manifolds. In *An Introduction to Manifolds*, pages 47–83. Springer.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. (2017). Graph attention networks. *stat*, 1050:20.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2018). How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.

Yan, Y., Hashemi, M., Swersky, K., Yang, Y., and Koutra, D. (2021). Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. *arXiv preprint arXiv:2102.06462*.

Zhao, L. and Akoglu, L. (2019). Pairnorm: Tackling oversmoothing in gnns. *arXiv preprint arXiv:1909.12223*.

Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., and Koutra, D. (2020). Generalizing graph neural networks beyond homophily. *arXiv preprint arXiv:2006.11468*.

APPENDIX A
MATHEMATICAL MATERIAL

In this section we give further mathematical intuition on the topological nature of graphs and supplementary proofs which did not make it in the main chapters.

A.1 Graphs and Topology

Given a set X, a common question one may ask is how close together are two points $x, y \in X$. In a *metric space* (X, d), this is clearly answered via its corresponding *metric* $d : X \times X \to [0, \infty)$ which takes two elements from the set and outputs a real number which numerically quantifies their closeness.

In some sense topological spaces generalise this notion. They define the concept of closeness in a more abstract way, through the use of a topology $\mathcal{T}$ and its corresponding elements.

Definition A.1.1. (Kelley, 2017) A topology $\mathcal{T}$ over a set X is a collection of of subsets of X called open sets, which follow the following axioms:

1. The empty set $\emptyset$ and X are in $\mathcal{T}$.
2. Any (potentially infinite) union of elements of $\mathcal{T}$ is in $\mathcal{T}$.
3. Any finite intersection of elements of $\mathcal{T}$ is in $\mathcal{T}$.

Topological spaces are fundamental because they are the most general type of space which allow for the definition of limits, continuity and connectedness (Kelley, 2017). Graphs are usually thought of as combinatorial objects, but they are topological objects in their own right. In our work we are particularly interested in cellular complexes, a type of topological space which is very popular in the field of algebraic topology.

Definition A.1.2. (Hansen and Ghrist, 2019) A (regular) cell complex is a topological space X together with a partition $\{X_\sigma\}_{\sigma \in P_X}$ of subspaces X_σ of X called cells such that:

1. For each element $x \in X$ there exists an open neighbourhood of x that intersects finitely many cells.
2. For all σ, τ we have that $X_\tau \cap \overline{X_\sigma} \neq \emptyset$ iff $X_\tau \subseteq \overline{X_\sigma}$, where $\overline{X_\sigma}$ is the closure of a cell.
3. Every cell is homeomorphic to $\mathbb{R}^n$.
4. (Regularity) For every $\sigma \in P_X$ there is a homeomorphism ϕ of a closed ball in $\mathbb{R}^{n_\sigma}$ to $\overline{X_\sigma}$ such that the restriction of ϕ to the interior of the ball is a homeomorphism onto X_σ.

While the definition of a cell complex is rather technical, the conditions just guarantee that the structure of the cells is what one would expect. A cell complex is a hierarchical structure made of progressively higher dimensional cells.

One starts with a set of vertices which correspond to 0-cells. To form 1-cells, one connects vertices together with line segments. If we stop here, then we have effectively constructed a (multi) graph, where vertices are 0-cells and edges are 1-cells. In other words, a graph is a 1-dimensional cellular complex. Cellular complexes may have more structure than a graph, for example 2-cells may be added by attaching a circle to a cycle in the graph. Generally, an n-cell may be added by attaching an n-dimensional sphere to certain $(n-1)$-cells in the cellular complex.

While this may sound very abstract, it has actually been shown to be useful when building GNNs. Traditional message passing operates up to 1-cell structures. Bodnar et al. (2021a) show that one can build message-passing layers which are aware of cells of higher dimensions and that this is particularly beneficial in detecting cycles, with important applications to, for example, chemistry. In our work we make the simplification to work over 1-dimensional cellular complexes, however cellular sheaf theory is perfectly well equipped for higher cellular dimensions hinting to a possible exciting research direction.

A.2 Supplementary Proofs

We give some supplementary proofs. First we show the convergence of the sheaf heat equation, from theorem 4.1 by Hansen and Ghrist (2021).

Theorem A.2.1. (Hansen and Ghrist, 2021) Solutions $\mathbf{x}(t)$ of the sheaf heat equation (2.22), in the limit $t \to \infty$ converge to the orthogonal projection of the starting signal $\mathbf{x}(0)$ onto $H^0(\mathcal{G}, \mathcal{F})$.

Proof. The sheaf Laplacian $\mathbf{L}_\mathcal{F}$ is a symmetric positive semi-definite matrix as $\mathbf{x}^T \mathbf{L}_\mathcal{F} \mathbf{x} = \mathbf{x}^T \delta^T \delta \mathbf{x} = (\delta \mathbf{x})^T \delta \mathbf{x} \geq 0$. As a consequence the matrix has orthogonal eigenbasis with all eigenvalues $\lambda \geq 0$.
The solution of the sheaf heat equation (2.22) is $\mathbf{x}(t) = \exp(-t\alpha \mathbf{L}_\mathcal{F})\mathbf{x}(0)$. If we work under the orthogonal eigenbasis, the solution operator is a diagonal matrix with entries $\exp(-t\lambda)$ for each eigenvalue λ of $\mathbf{L}_\mathcal{F}$. As $t \to \infty$ we have that for $\lambda > 0$, $\exp(-t\lambda) \to 0$. Instead for $\lambda = 0$, $\exp(-t\lambda) \to 1$. In particular the 0 eigenvalues correspond to the kernel of $\mathbf{L}_\mathcal{F}$.
This means that as $t \to \infty$, the solution is an orthogonal projection onto $\ker \mathbf{L}_\mathcal{F}$ and by the Hodge theorem $\ker \mathbf{L}_\mathcal{F} = H^0(\mathcal{G}, \mathcal{F})$. □

We show the convergence result of GRAND from Di Giovanni et al. (2022).

Lemma A.2.2. (Di Giovanni et al., 2022) The GRAND dynamical system:

$$\frac{\partial \mathbf{X}(t)}{\partial t} = -\left(\mathbf{I} - \Lambda \mathbf{I}\right) \mathbf{X}(t) \tag{A.1}$$

converges to the mean vector $\boldsymbol{\mu}$ of the intial features.

Proof. We can write the solution of the GRAND dynamical system in its Jordan form:

$$\mathbf{X}(t) = \mathbf{P}\mathbf{J}\mathbf{P}^{-1}\mathbf{X}(0) \tag{A.2}$$

for some invertible matrix $\mathbf{P}$ of eigenvectors and where $\mathbf{J}$ is a diagonal matrix with diagonal entry proportional to $\exp(-(1-\lambda)t)$ for each eigenvalue λ of the matrix Λ.
By assumption, the underlying graph G is connected and as a consequence the attention matrix Λ is *regular*. As a consequence, by the Perron–Frobenius theorem, we have that for every eigenvalue $\mathrm{Re}(\lambda)$ is smaller than 1, except with the largest eigenvalue being 1 associated to the Perron–Frobenius eigenvector $\mathbf{1}$. As $t \to \infty$ we have that for $\lambda < 1$, $\exp(-(1-\lambda)t) \to 0$, while when $\lambda = 1$ we have that $\exp(-(1-\lambda)t) \to 1$.
This acts as an orthogonal projection of $\mathbf{X}(0)$ onto the Perron-Forbenius eigenvector which is simply the mean vector $\boldsymbol{\mu}$ over the initial features $\mathbf{X}(0)$. □

APPENDIX B
EXPERIMENTS

B.1 Datasets

In this section we summarise and describe the real-world and synthetic datasets used throughout the experiments.

B.1.1 Real-World

What follows is a description of the real-world datasets. The descriptions are summarised from the PyTorch Geometric documentation.

WebKB Datasets include Cornell, Texas and Wisconsin. The nodes represent web pages and edges represent hyperlinks between them. The features are bag-of-words representations of the web-pages. The task is to classify nodes into 5 categories: student, project, course, staff, and faculty.

WikipediaNetwork Datasets include Cornell and Chameleon. The nodes represent web pages and edges represent hyperlinks between them. The features are informative nouns in the Wikipedia pages. The task is to predict the daily traffic of the web pages.

Actor Film dataset. The nodes represent actors and edges represent co-occurence on the same Wikipedia page. Node features correspond to keywords in the Wikipedia pages. The task is to classify nodes into five categories.

Planetoid Datasets include Cora, Citeseer and Pubmed. The graphs represent citation networks. Nodes represent documents and edges represent citations between them.

B.2 Synthetic

To generate the synthetic graphs, we first generate n nodes belonging to 3 different classes (with equal priors for each class). Each node feature vector $\mathbf{x}$ is 3-dimensional with features sampled from Gaussian distributions. $\mathcal{N}(\mu, \sigma^2)$, such that $\mathbf{x}_1$ is sampled from $\mathcal{N}(-0.6, 0.4)$, $\mathbf{x}_2$ from $\mathcal{N}(0, 0.4)$ and $\mathbf{x}_3$ from $\mathcal{N}(0.6, 0.4)$. The $|E|$ edges are then sampled such that with probability h they connect two nodes of the same class and probability $1-h$ they connect two nodes of different classes. Figure B.1 shows the spectrum of synthetically-generated graphs used in our experiments.

Remark. When we generate the graphs, we still maintain the constraint that the graphs must be connected. For this reason $h = 1$ is not possible as this would always produce a graph with 3 disconnected sub-graphs. For this reason our highest h is set to 0.9.

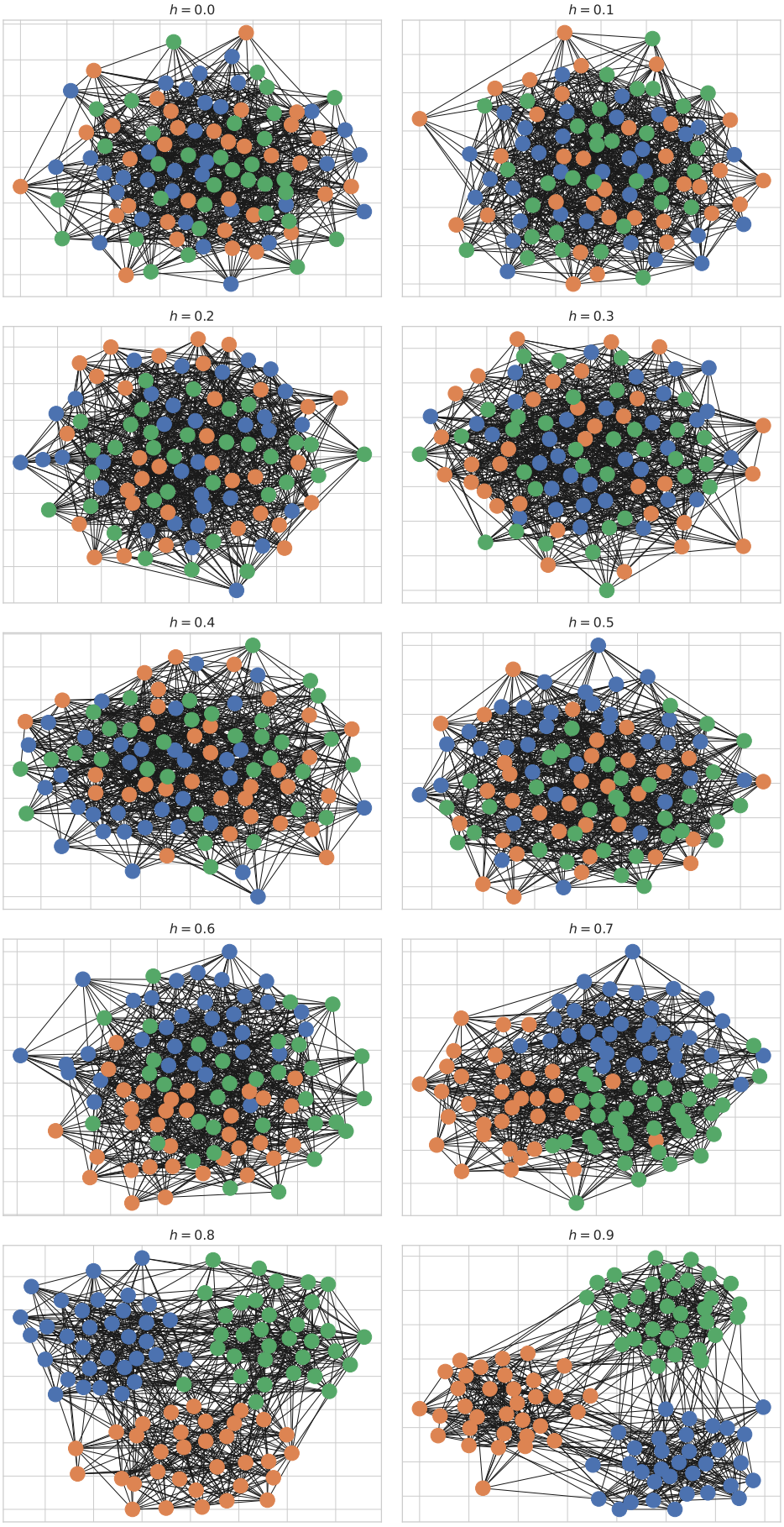


Fig. B.1 Spectrum of generated graphs with different homophily levels. As h increases, the classes become more evidently separated.

B.3 Hyper-parameters

Table B.1 details the hyper-parameter search space used in all of our experiments. The reported test-set accuracy is then chosen to be the one with the tuned model reporting the highest validation set accuracy. We train until the maximum number of epochs have been reached and do early stopping only if the validation performance does not improve for a number of a patience epochs. The hyper-parameter search space is very similar to the one used by Bodnar et al. (2022).

Table B.1 Hyper-parameter search space for SAN and ANSD.

Hyper-parameter	Searchable Values
Hidden channels	$\{8, 16, 32\}$ (WebKB) and $\{8, 16, 32, 64\}$ (others)
Stalk width d	$\{1, 2, \ldots, 5\}$
Layers	$\{1, 2, \ldots, 8\}$
Learning rate	0.02 (WebKB) and 0.01 (others)
Activation	ELU
Regular weight decay	Log-uniform $[-4.5, 11.0]$
Sheaf weight decay	Log-uniform $[-4.5, 11.0]$
Layer dropout	Uniform $[0, 0.9]$
Patience (epochs)	100 (Wiki) and 200 (others)
Max training epochs	1000 (Wiki) and 500 (others)
Optimiser	Adam (Kingma and Ba, 2014) ($\beta_1 = 0.9$, $\beta_2 = 0.999$)
Attention Heads	8