Sparse & Interpretable Graph Attention Networks

Titus Naber

Delft University of Technology

Sparse & Interpretable Graph Attention Networks

by

Titus Naber

Student Name	Student Number
Titus Naber	4601394

to obtain the degree of Master in Computer Science with Specialization in Artificial Intelligence at Delft University of Technology,
to be publicly defended on Wednesday, October 11th 2023 at 13:00

Thesis Supervisor:	Dr. E. Isufi, TU Delft
External Supervisor:	Dr. M. V. Treviso, Instituto Superior Técnico
Committee Members:	Dr. E. Isufi, TU Delft (Chair)
	Dr. M. V. Treviso, Instituto Superior Técnico
	Dr. J.L. Cremer, TU Delft
	Dr. M. Khosla, TU Delft
Faculty:	Multimedia Computing Group, Facutly of EEMCS, TU Delft

Preface

While this work marks the end of my formal education at TU Delft, my journey as a lifelong learner is boundless. Reflecting on the past year, starting with a broken arm and a blank slate in graph learning, I have transformed into a knowledgeable individual well-versed in this domain, and luckily with a fully functioning arm. Such an endeavour is not possible alone; it was made possible through the collective support and guidance I received.

First and foremost, I extend my gratitude to Marcos for his dedication and the countless hours invested in this thesis. A mere year ago, we embarked on this journey as newcomers to the field of graph learning. Through numerous meetings and stimulating discussions, we not only gained profound insights but also collaborated on a paper that delves into this very subject. In addition, I must express my sincere appreciation to Elvin for devoting his time and effort to this thesis and his dedication to the completion of the paper. His open-door policy and invaluable feedback have significantly shaped the outcome of this work. Furthermore, I would like to thank Dr. André Martins for adding his expertise in the final stages of the paper and would also like to acknowledge Dr. Jochem Cremer and Dr. Megha Khosla for being part of my thesis committee.

Next, I want to extend my appreciation to my friends and family, whose support made this project a reality. To my wonderful roommates, who made studying at home an experience full of joy and laughter. To Lisa and Casper, for countless study sessions and the cherished tea breaks that kept us going. Lastly, I am thankful to my parents who, maybe without fully grasping the subject, have stood by me throughout this demanding journey.

To a lifetime of learning,
Titus Naber
Delft, October 2023

Summary

In this thesis, we study the impact of sparsity on both the performance and interpretability of Graph Attention Networks. Additionally, we introduce two novel methods that yield Sparse & Interpretable Graph Attention Networks.
In Chapter 1, we introduce the reader to the concept of Graph Attention Networks (GATs), sparse attention, and interpretability. Subsequently, we provide our research motivation and formulate the research question.
In Chapter 2, the necessary background information is presented at a fundamental level. First, the reader is formally introduced to Graph Neural Networks (GNNs) and GATs. Continuing, various approaches to sparse attention are discussed in detail. Finally, an overview of the current approaches to explainability in GNNs is provided, along with a focus on the category that the methods in this research can be assigned to.
Chapter 3 consists of the paper written on this topic. As this report acts as an envelope of the paper, the introduction and background of the paper overlap with the corresponding chapters in this report, although the paper is written more formally and concisely. Most importantly, this chapter contains an in-depth explanation of the methods developed in this study and their evaluation. Furthermore, rigorous evaluations are performed and presented in the form of Pareto curves, providing insights into the performance-interpretability trade-off for all datasets. Finally, an appendix is provided containing details regarding the evaluation and some additional results.
The paper has to function as a stand-alone research and is therefore considered the core of this report. However, a paper has its limitations due to its concise nature. Thus, Chapter 4 contains additional evaluations performed to gain insight into the behaviour of sparsity within the proposed methods and the effect of changing the sparsity parameter after training. Furthermore, we presented a failed concept due to its relevance for future work. Additional related work is presented in Chapter 5, where we discuss the idea of attention as an explanation and present other self-interpretable methods within the field.
This research is concluded by providing an answer to the research question in Chapter 6, along with suggestions for future work.

Contents

1 Introduction

In our increasingly data-centric world, networks have become an essential tool for representing and comprehending complex relationships among numerous entities. Without us even realizing it, networks form an integral part of our surroundings, encompassing the expansive networks of railroads (Figure 1.1) to the visualizations of molecule structures. In the last decades, networks – also referred to as graphs – have also evolved into an unmissable source of information for a wide-ranging variety of computational tasks, leading breakthroughs in fields such as social network analysis and biological computing [2, 24].

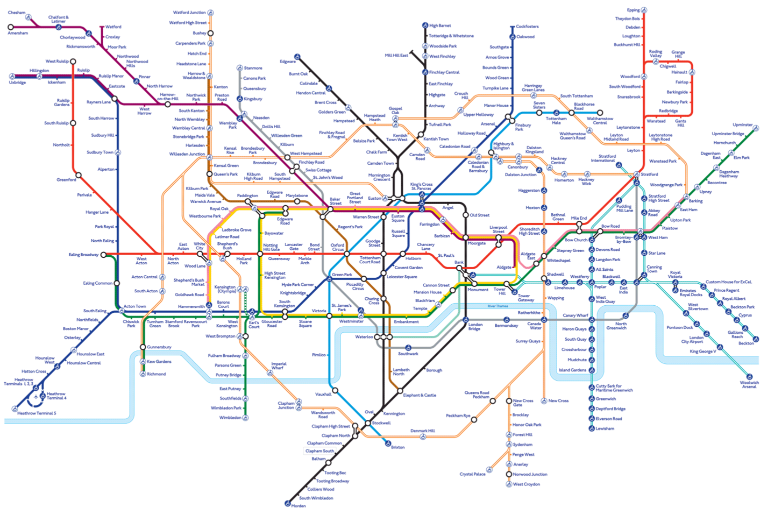


Figure 1.1: A map of the London Underground, a network/graph where nodes represent stations and edges the lines in between [16].

With the advent of machine learning revolutionizing data processing, the Graph Neural Network (GNN) [29] has emerged as a powerful tool, enabling the extraction of even more valuable insights from graphs. This approach is unique in its ability to consider both the structure of a graph and the information provided by individual nodes in its predictions. Suppose we are examining a rail network, with nodes representing various cities and edges indicating direct routes. Employing a GNN, we can easily identify the most frequented tourist locations by considering information like the number of museums in each city, as well as structural components such as the number of direct routes leading to them (Figure 1.2a). However, the GNN is not optimal, determining the most popular sport at a specific location might pose a challenge due to a multitude of connections that are irrelevant to the classification. To address this, we could narrow our focus to only those nodes representing surrounding places that have a large sports stadium, illustrated in Figure 1.2b.

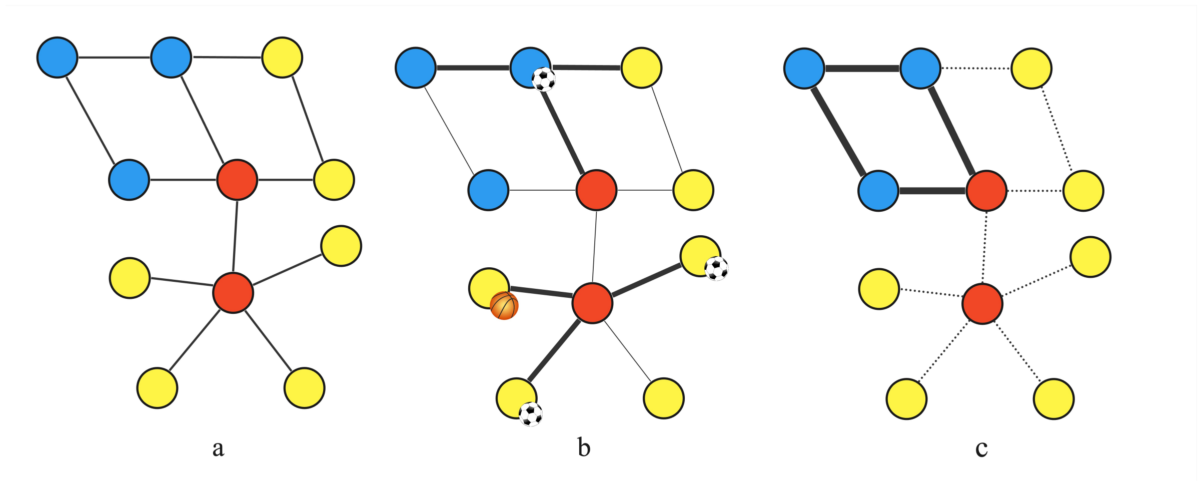


Figure 1.2: Illustration of an undirected graph representing railroad connections between cities. Blue nodes indicate culturally interesting places, red nodes large cities and yellow nodes indicate smaller places. The thickness of a line indicates the magnitude of the attention weights and a dashed line represents an attention weight of 0. In (a), we can identify the most populous cities based on their high connectivity. (b) provides an instance where most attention is directed towards nodes with sports stadiums, enabling us to find the most popular sport within the neighbouring area. Lastly, (c) demonstrates how a single structure suffices to determine whether a node qualifies as part of a travel route.

We refer to this concept of focusing on specific nodes in the neighbourhood as *attention* [38], where a high attention weight represents a high focus, producing the Graph Attention Network (GAT) [39]. The Graph Attention Network is an enhancement of the GNN that learns the importance of a node to the classification. Specifically, each node will assign attention weights to each of its neighbours, the total sum of these weights is 1, as illustrated in Figure 1.3. Intuitively, we can use these weights as a method to explain model decisions, e.g. we identify football as the most popular sport as we mainly attend to multiple places that have a football stadium. We refer to a model that can provide explanations for its decision as an interpretable model.

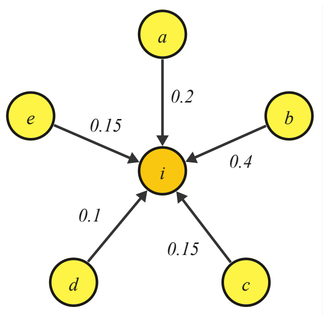


Figure 1.3: Example of the normalized attention weights distributed to each neighbour of node i. The impact of a neighbouring node on the classification of node i is scaled by the corresponding attention weight.

However, leveraging the attention weights to explain model decisions is not always optimal as it has two limitations. Firstly, due to the nature of the attention mechanism each node will always receive some attention, even though it is possible for those nodes not to be relevant to the classification at all. We refer to this as a dense attention distribution, assigning only non-zero values. Secondly, all attention weights are forced to sum to 1, not allowing a node to remove all its connections to neighbouring nodes. When formulating this in an example, we might try to predict whether a city is part of a travel route or not, a route where people travel to multiple cities by train. To be part of a route, a city or village would have to be an interesting tourist location and should be connected to other interesting locations. Ideally, if a node is part of a route, the network would return an explanation in such a way that only those edges that are part of the potential travel route are highlighted, as illustrated in Figure 1.2c. In the traditional setup, this is not possible.

To address the limitations posed by the conventional dense attention mechanism, we turn to the concept of sparse attention. By embracing sparse attention, we can assign zero values as attention weights, producing a sparse attention distribution. Thus, gaining the ability to exclude nodes that bear no relevance to the classification process and the flexibility for a node to sever all connections with its neighbouring nodes.

In current literature several sparse approaches exist focussing on increased robustness [15, 17], performance [28, 43], computation speed [33], scalability [32] or interpretability [27]. In this report, we focus on leveraging sparse attention to provide more interpretable networks, providing the following research question:

RQ. **How can we produce interpretable Graph Attention Networks through sparse attention?**

To answer this research question, we conduct extensive experiments on a diverse set of six datasets, comprising five real-world datasets and one synthetic dataset. We comprehensively assess the performance of three novel methods alongside five established baselines. Our key contributions are summarized as follows:

1. We propose *MapSelect*, a novel approach enabling precise control over the sparsity of attention layers, both locally and globally, leading to superior interpretability results compared to robust baseline methods.
2. We provide an extensive and unique evaluation of sparse GATs, examining trade-offs between sparsity-accuracy and sparsity-interpretability, and providing insights into architectural choices that influence interpretability.

Report structure This report is structured to envelop the main paper dedicated to the subject. The paper, provided in Chapter 3, stands alone. For those unfamiliar with the subject, a more fundamental background is provided in Chapter 2, others are recommended to read the paper directly. Chapter 4 will present additional experiments that were performed in this research and Chapter 5 will expand on the related work presented in the paper. In Chapter 6 a thesis summary and the answer to the research question are provided.

2
Background

This chapter provides the preliminary knowledge required of the reader. Recognizing that Chapter 3 assumes a certain level of familiarity with the surrounding context, this chapter will gradually establish the necessary background knowledge, serving as a substitute for the background section in the paper.

2.1. Graph Learning

2.1.1. Graphs

A *graph*, denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, is a mathematical structure used to represent relationships between entities. The set of nodes $\mathcal{V} = \{1, \ldots, N\}$ represents these entities, which could be diverse entities such as individuals, objects, or abstract concepts. The set of edges $\mathcal{E} = \{(i, j) \mid i, j \in \mathcal{V}\}$ connects pairs of nodes and captures the interactions or associations between them. Specifically, for a node $i \in \mathcal{V}$, its neighbours are denoted as $\mathcal{N}_i = j \in \mathcal{V} \mid (j, i) \in \mathcal{E}$, showcasing the connections it has within the graph. A graph that is fully contained by another graph is referred to as a subgraph and can be denoted as $\mathcal{G}' \subseteq \mathcal{G}$.

To encode the structure of the graph, the adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ is introduced. In this matrix, an entry $A_{ij} = 1$ signifies the existence of an edge from node i to node j, while $A_{ij} = 0$ indicates no connection. Such a connection can be *directed* or *undirected*, indicating whether the edge denotes a one-way or two-way relation. Notably, for the undirected case, the adjacency matrix is symmetrical. To allow for self-referential relationships, the adjacency matrix can be augmented with self-loops, forming the modified adjacency matrix $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}_N$, where $\boldsymbol{I}_N \in \mathbb{R}^{N \times N}$ represents an identity matrix. A feature vector belonging to node i is denoted as $\boldsymbol{h}_i$.

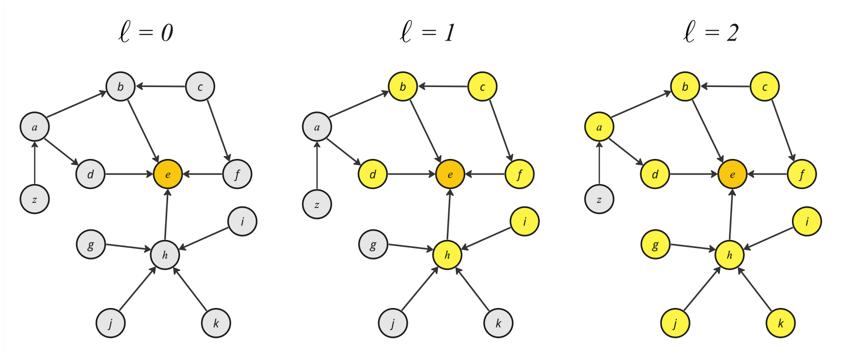


Figure 2.1: Illustration of two iterations of the message passing-passing function with target node e. The yellow and orange nodes represent the computation graph for the features of node e. According to Equation 2.1, at $\ell = 2$, h_e^3 will be a vector containing the features of all yellow nodes and node e itself, allowing node e to be classified based on its own features and those of its neighbours. Messages are passed over the direction of the edge and self-loops are not included.

2.1.2. Graph Neural Networks

The *Graph Neural Network* (GNN) [29] is a powerful deep learning method designed for classifying nodes and graphs. It does so by leveraging both node features and the underlying graph structure. Notably, GNN exhibits robustness to variations in the size of the input graph, whether in terms of nodes or edges. A GNN is implemented by combining the message-passing algorithm with learnable functions. Message-passing is a fundamental concept in graph learning, allowing information exchange between nodes within a graph. Each node aggregates and updates its own features based on messages received from neighbouring nodes. Mathematically, for a node $i \in \mathcal{V}$, the updated feature representation $\boldsymbol{h}_i^{(\ell+1)}$ after passing the input features through a GNN layer can be expressed as:

$$\mathbf{h}_i^{(\ell+1)} = f_\phi^{(\ell)}\left(\mathbf{h}_i^{(\ell)}, \bigoplus_{j \in \mathcal{N}(i)} g_\phi^{(\ell)}\left(\mathbf{h}_i^{(\ell)}, \mathbf{h}_j^{(\ell)}\right)\right). \tag{2.1}$$

Here, $\mathcal{N}_i$ represents the neighbours of node i, $\boldsymbol{h}_i^{(\ell)}$ denotes the initial feature representation of node i, ℓ denotes the current layer, and $f_\theta : \mathbb{R}^d \to \mathbb{R}^{d'}$ and $g_\phi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ represent learnable functions with an output dimension of d' and input dimension of d. The aggregation, signified by $\bigoplus$, combines the features of all neighbouring nodes. Common aggregation functions include taking the sum, mean, or maximum of the vector representations, leading to different architectures [11, 15].
The message-passing function is implemented by the aggregation of all neighbours $\mathcal{N}_i$ at each node i and can be performed iteratively to incorporate more neighbouring feature vectors in h_i. In the initial iteration, a node gathers features from its neighbours. In the subsequent iteration, this process is repeated, now with each neighbour having already collected features from their own neighbours in the first iteration, ultimately passing this collective information to the first node. Thus, each iteration increases the distance to a neighbour by one. This distance is referred to as the *k-hop distance* (depicted in Figure 2.1) and represented by ℓ in Equation 2.1. All nodes within the k-hop distance to the target node i are included in the computation of $\boldsymbol{h}_i^{(\ell)}$, referred to as the *computation graph*.

2.1.3. Graph Attention Networks

A *Graph Attention Network* (GAT) [39] extends the GNN with an attention mechanism [38]. This attention mechanism will determine the importance of neighbouring nodes to the classification of the target node. During the aggregation step, the features of the neighbours will be scaled according to their importance. Scaling these features creates a more focused feature vector, making classification easier. In Figure 1.3 an example of the weighted neighbourhood of a node is illustrated.
More formally, given a set of node representations $\{\boldsymbol{h}_i^{(0)} \in \mathbb{R}^d \mid i \in \mathcal{V}\}$ as input (at layer $\ell = 0$), a GAT layer first computes *attention scores* for edges $(i, j) \in \mathcal{E}$ as follows:

$$z_{ij}^{(\ell)} = \boldsymbol{a}^{(\ell)} \cdot \text{LeakyReLU}\left(\boldsymbol{W}^{(\ell)} \cdot \text{concat}(\boldsymbol{h}_i^{(\ell)}, \boldsymbol{h}_j^{(\ell)})\right), \tag{2.2}$$

where $\boldsymbol{a}^{(\ell)} \in \mathbb{R}^{d'}$, $\boldsymbol{W}^{(\ell)} \in \mathbb{R}^{d' \times 2d}$ are learnable linear transformations, concat$(\cdot)$ denotes vector concatenation, and the non-linear LeakyReLU activation function is applied. In contrast to a traditional ReLU, where $f(x) = \max(0, x)$, the LeakyReLU allows a small positive gradient when the unit is not active. *Attention weights* are then obtained by employing the *softmax* transformation:

$$\pi_{ij}^{(\ell)} = \text{softmax}_j(\boldsymbol{z}_i^{(\ell)}) := \frac{\exp(z_{ij}^{(\ell)})}{\sum_{j \in \mathcal{N}_i} \exp(z_{ij}^{(\ell)})}, \tag{2.3}$$

where $\boldsymbol{z}_i^{(\ell)} \in \mathbb{R}^n$ represents the attention scores of node i, with $n = |\mathcal{N}_i|$. Softmax is a transformation function that maps scores to a probability distribution such that all input variables will receive some probability mass, as plotted in Figure 2.4. Note that, we refer to attention scores (z_{ij}) before Equation 2.3 and to attention weights (π_{ij}) after this normalization.
Finally, the new vector representation $\boldsymbol{h}_i^{(\ell+1)} \in \mathbb{R}^{d'}$ for node i is determined by the weighted average

of the transformed features from neighbouring nodes, potentially followed by a non-linear pointwise function σ:

$$h_i^{(\ell+1)} = \sigma\left(\sum_{j \in \mathcal{N}_i} \pi_{ij}^{(\ell)} W^{(\ell)} h_j^{(\ell)}\right), \tag{2.4}$$

where $W^{(\ell)} \in \mathbb{R}^{d' \times d}$ is a learnable linear transformation. It is important to note that in this notation, aggregation is performed by summing the features.

2.2. Sparse Attention

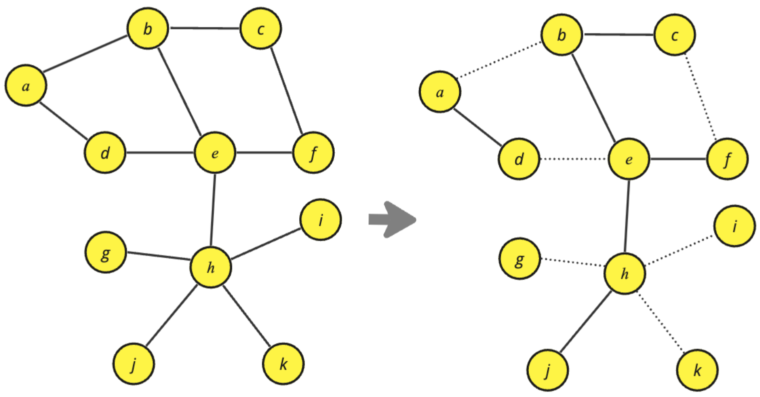


Figure 2.2: A sparse subgraph is produced by masking out edges in the original graph.

2.2.1. Sparse Subgraphs

In the preceding section, we established a comprehensive understanding of graphs and their role in defining relationships. Naturally, not every relation within a graph is equally important, leading to the attention mechanism provided by a GAT. Following this trajectory, one can explore the possibility of selectively eliminating specific edges from the original graph, creating a *sparse subgraph*, a subset of edges of the original graph that only contains the most important relations (Figure 2.2). In this context, sparsity refers to the property of a graph having fewer edges than the original graph. Formally, a subgraph $\mathcal{G}' \subseteq \mathcal{G}$ is referred to as a sparse subgraph when:

$$\frac{|\mathcal{E}'|}{|\mathcal{V}'|} \ll \frac{|\mathcal{E}|}{|\mathcal{V}|}. \tag{2.5}$$

Conversely, when referring to a graph that has more edges than its original or a different graph, we refer to this graph as more *dense*.

2.2.2. α-entmax

One approach to generating a sparse subgraph is to transform the attention scores in such a way that irrelevant edges will not be attended to at all, receiving a probability mass of zero. Commonly, the softmax function is used to normalize the attention scores within a neighbourhood to provide a stable convergence. However, the softmax function will always attribute some probability mass to every value, no matter how small (Figure 2.4). In [18], an alternative transformation is proposed called *sparsemax*, which outputs a sparse distribution over the input while preserving the necessary differentiability for back-propagation. The work in [25] proposes, *α-entmax*, an enhancement of sparsemax that is controllable via the α parameter. To fully understand these methods we first need to take a closer look at the standard softmax function.

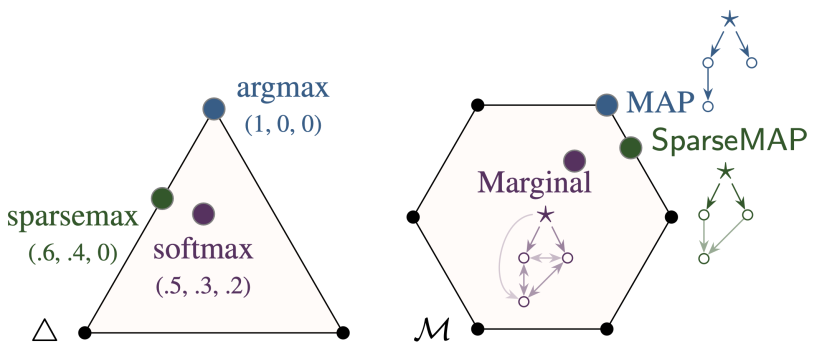


Figure 2.3: An illustration of the search space for traditional inference (left) and structured inference (right). The search space of α-entmax is defined by the simplex $\triangle$, enforcing a probability distribution. Structured inference is constrained by its polytype $\mathcal{M}$, which defines the various potential structure assignments that present a valid solution to the optimization problem. Optimal solutions will be on the edge of the search space whereas more nuanced solutions move closer to the centre, the sparse solutions SparseMAP and sparsemax are in between. [23]

The goal of a probability transformation is to map a given input $z \in \mathbb{R}^d$ onto the *probability simplex* $\triangle^{d-1}$ in a differentiable manner. The simplex is a geometric object such that vectors in it are constrained to be positive and sum to one, denoted as $\triangle^{d-1} := \{\boldsymbol{\pi} \in \mathbb{R}^d \mid \boldsymbol{\pi} \geq \mathbf{0},\ \mathbf{1}^\top \boldsymbol{\pi} = 1\}$ and illustrated in Figure 2.3. This forces the function to output a probability distribution. The simplest solution in the simplex is produced by simply selecting the highest values in z, denoted as:

$$\arg\max(z) := \arg\max_{\boldsymbol{\pi} \in \triangle^{d-1}} z^\top \boldsymbol{\pi}, \tag{2.6}$$

where z is the input and $\boldsymbol{\pi}$ is the solution. By the Fundamental Theorem of Linear Programming [3], the solution will lie on a corner of the probability simplex, and therefore $\boldsymbol{\pi}$ will be a binary vector with a 1 at the component of the highest value in z, as illustrated in Figure 2.3.
The original GAT leverages softmax to map z onto the simplex (Equation 2.3), which can also be written in the variational form [40],

$$\text{softmax}(z) := \arg\max_{\boldsymbol{\pi} \in \triangle^{d-1}} z^\top \boldsymbol{\pi} + \mathrm{H}^{\mathrm{S}}(\boldsymbol{\pi}). \tag{2.7}$$

where $\mathrm{H}^{\mathrm{S}}(\boldsymbol{\pi}) := -\sum_j \pi_j \ln \pi_j$ is the Gibz-Boltmann-Shannon entropy. Using the variational form we can change the entropy to change the characteristics of the transformation.

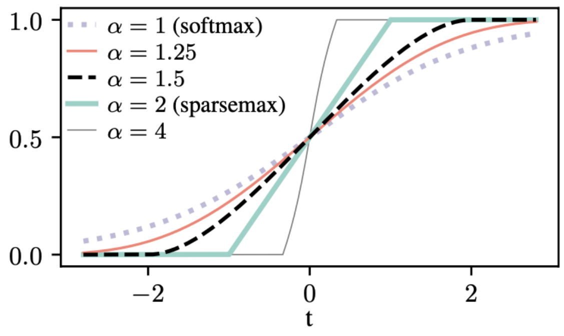


Figure 2.4: The α-entmax function mapped onto domain t. Using the α parameter, the behaviour of Sparsemax and softmax can be recreated. [25]

Sparsemax The sparsemax transformation [18] implements the Gini entropy: $\mathrm{H}^{\mathrm{G}}(\boldsymbol{\pi}) := \frac{1}{2}\sum_j \pi_j \left(1 - \pi_j\right)$. This yields a sparse probability distribution and can be written as

$$\begin{aligned}\text{sparsemax}(\boldsymbol{z}) &:= \underset{\boldsymbol{\pi}\in\Delta^{d-1}}{\arg\max}\, \boldsymbol{z}^\top\boldsymbol{\pi} + \mathrm{H}^{\mathrm{G}}(\boldsymbol{\pi}) &(2.8)\\ &:= \underset{\boldsymbol{\pi}\in\Delta^{d-1}}{\arg\min}\, \|\boldsymbol{\pi} - \boldsymbol{z}\|^2. &(2.9)\end{aligned}$$

This transformation is likely to assign exactly zero probability to low-scoring edges allowing for a more focused attention map than created by the softmax function. Yet, it is impossible to control this function; therefore, α-entmax is introduced next.

Entmax α-entmax [25] leverages the Tsallis entropy [36], introducing control over the zero thresholds through the α parameter, defined as:

$$H_\alpha^T(\boldsymbol{\pi}) := \begin{cases} \frac{1}{\alpha(\alpha-1)}\sum_j(\pi_j - \pi_j^\alpha), & \alpha \neq 1 \\ -\sum_j \pi_j \log \pi_j, & \alpha = 1, \end{cases} \tag{2.10}$$

leading to,

$$\alpha\text{-entmax}(\boldsymbol{z}) := \underset{\boldsymbol{\pi}\in\triangle^n}{\arg\max}\, \boldsymbol{z}^\top\boldsymbol{\pi} + H_\alpha^T(\boldsymbol{\pi}). \tag{2.11}$$

The solution to the above optimization problem is:

$$\alpha\text{-entmax}(\boldsymbol{z}) = [(\alpha-1)\boldsymbol{z} - \tau(\boldsymbol{z})\mathbf{1}]_+^{\frac{1}{\alpha-1}}, \tag{2.12}$$

where $[\cdot]_+$ is the positive part (ReLU) function, and $\tau : \mathbb{R}^n \to \mathbb{R}$ is a normalizing function satisfying $\sum_j \left[(\alpha-1)z_j - \tau(\boldsymbol{z})\right]_+^{\frac{1}{\alpha-1}} = 1$ for any $\boldsymbol{z}$. As a result, entries with score $z_j \leq \frac{\tau(\boldsymbol{z})}{\alpha-1}$ get exact zero probability and can be disregarded during computation.
The value of α controls the propensity of sparsity. In the limit $\alpha \to 1$, α-entmax recovers the softmax function, while for increasing values of $\alpha > 1$ the output distribution becomes more sparse. In particular, we recover sparsemax with $\alpha = 2$. The propensity for different settings of α is displayed in Figure 2.4.

Both sparsemax and α-entmax offer efficient and differentiable algorithms, making them seamless alternatives to the softmax function. Although α-entmax provides some control over the sparsity, it does not offer precise control over the size of the resulting subgraph, leaving room for further improvement.

2.2.3. SparseMAP

In this section, we review *SparseMAP*, a sparse structured inference method that offers precise control. For situations where one wants to specify precisely the number of neighbours, a top-k operation can also be used, such that only the k largest entries in $\boldsymbol{\pi}$ are kept, while the others are zeroed out [9]. However, the top-k operation is not differentiable and may inevitably introduce instabilities for training.
SparseMAP addresses this issue by casting subset selection as a relaxed structured prediction problem. Structured inference is the practice of finding the highest-scoring structure that satisfies a set of given constraints. Although this is challenging as the search space $\mathcal{S}$ of a structured problem often grows exponentially (e.g., all n-length binary sequences $\{0,1\}^n$), it does offer the advantage of accommodating a wide array of constraints. Specifically, when considering n-length binary sequences with at most B non-zeros, SparseMAP is defined as:

$$\begin{aligned}\text{SparseMAP}(\boldsymbol{z}; B) := &\underset{\boldsymbol{\mu}\in\mathcal{M}_\mathcal{S}}{\arg\max}\, \boldsymbol{z}^\top\boldsymbol{\mu} + \frac{1}{2}\|\boldsymbol{\mu}\|_2^2 \\ &\text{s.t.} \quad \mathcal{M}_\mathcal{S} := \Big\{\sum_i \pi_i \boldsymbol{y}_i : \boldsymbol{\pi} \in \triangle_{|\mathcal{S}|-1}, \boldsymbol{y}_i \in \mathcal{S}\Big\} \\ &\qquad \mathcal{S} := \{\boldsymbol{y} \in \{0,1\}^n \mid \|\boldsymbol{y}\|_1 \leq B\},\end{aligned} \tag{2.13}$$

where the cost function aims at finding a vector μ that is aligned with the scores z but with a quadratic regularizer to ensure smoothness. The solution is confined to the marginal polytope $\mathcal{M}_S$ that imposes solutions to be in the space of bit vectors $[0,1]^n$ with at most B non-zeros. Notably, the vertices of $\mathcal{M}_S$ represent binary solutions and in such cases we obtain $\boldsymbol{u} \in \{0,1\}^n$, whereas its edges represent a sparse convex combination of binary vectors, and thus $\boldsymbol{u} \in [0,1]^n$. Finally, the faces of this polytope lead to fully dense solutions. Because of the quadratic regularization term, SparseMAP promotes sparse vectors $\boldsymbol{u} \in [0,1]^n$ that lie on the boundary of the marginal polytope (vertices, edges, or other low-dimensional faces), as illustrated in Figure 2.3. SparseMAP is a technique that arises from the combination of Marginal inference and MAP inference. However, a detailed discussion of SparseMAP is beyond the scope of this report. For a more comprehensive understanding, interested readers are directed to [23, 22].

2.3. Explainability

A general limitation of deep learning is having *interpretable* model decisions. We refer to a decision as interpretable when we are able to explain or present it in understandable terms to a human [5]. Focusing on how interpretable a decision is, we can also define it as: "interpretability is the degree to which a human can understand the cause of a decision" [21].
As many fields in science, such as chemistry and computer security, attach great value to the ability to provide an explanation for a decision much research has been done in the field of *explainability*. Explainability with regard to graph neural networks has many approaches. A full taxonomy is shown in Figure 2.5. In this section, we will explore different methods to obtain model explanations and how to evaluate these explanations.

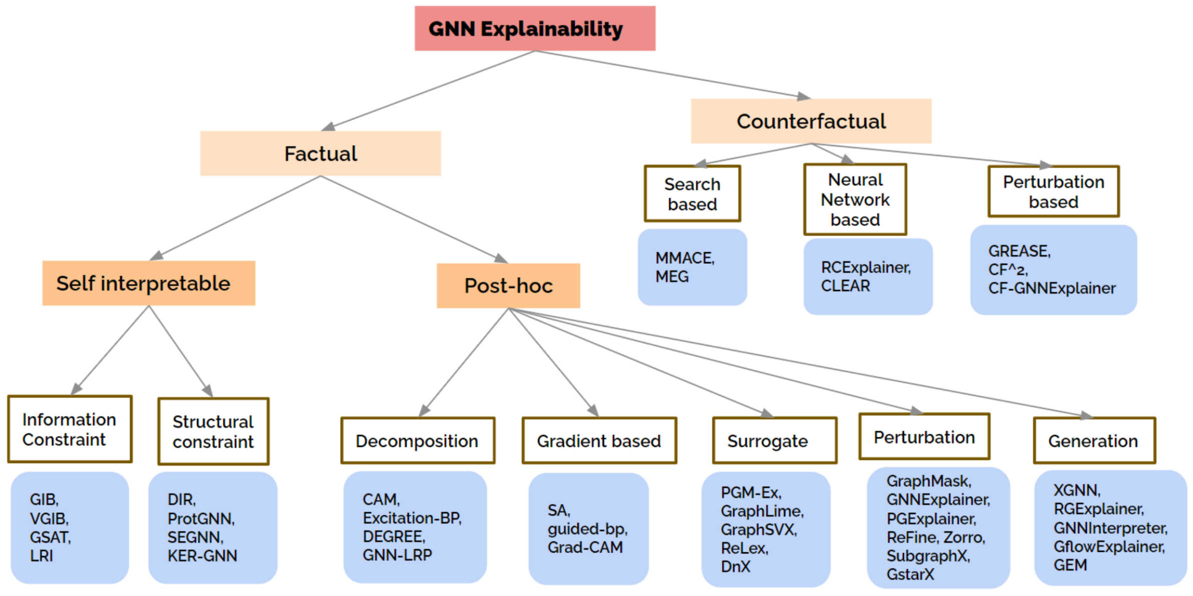


Figure 2.5: A taxonomy of the current approaches to explainability in GNNs as provided by [14]

2.3.1. Current Approaches to Explainability

Explainability can be divided into *factual* and *counterfactual* methods. Factual methods aim to find an explanation in the form of input features with the maximum influence over the prediction. Such an explanation is also referred to as a *rationale* and has the form of a set of features, indicating which features belonging to a node are relevant; or of a substructure defined by either edges or nodes, indicating what structural elements of the graph are relevant for the classification.
Counterfactual methods provide an explanation by finding the smallest change in the input graph that changes the model's prediction. Such an approach is beneficial for finding similar features that have a high influence on the outcome of the model.
In this research, the focus lies on factual explanations, e.g. the set illustrated in Figure 2.6. Factual approaches can be classified into two categories, *post-hoc* and *self-interpretable* methods.

Post-hoc methods

Post-hoc approaches involve generating explanations for model decisions after the model has been trained and made its predictions. These methods involve an additional model or algorithm that evaluates the GNN based on its input and output to generate an explanation. The following approaches make up the set of post-hoc methods:

- **Decomposition-based:** Decomposition-based methods consider the prediction of the model as a score that is decomposed and distributed backwards in a layer-by-layer fashion until it reaches the input. Backtracking this composition an explanation can be constructed based on the scores assigned as a result of specific parts of the input.
- **Gradient-based:** Gradient-based methods utilize gradient information to highlight the input features that had the most significant impact on a model's decision. A gradient can be viewed as a measure of the rate of change, and when applied to the prediction in relation to the input, it indicates the level of sensitivity of the prediction to variations in the input. This sensitivity is seen as a measure of importance.
- **Surrogate:** Surrogate methods involve training an interpretable model, such as a decision tree or linear regression, to approximate the behaviour of the underlying complex model. The interpretable surrogate model can then be used to provide explanations for the complex model's predictions.
- **Perturbation-based:** Perturbation-based methods involve introducing controlled changes to input features and observing the resulting alterations in predictions. By systematically perturbing inputs and monitoring the corresponding prediction changes, these methods uncover the model's response to different feature variations. The magnitude of the response to an input change defines the importance of the input.
- **Generation-based:** Generation-based methods use generative models or graph generators to derive instance-level or model-level explanations. Methods that are used for this generation include reinforcement learning and policy learning, often deployed to produce explanatory subgraphs.

Self-interpretable methods

Self-interpretable models are designed with transparency in mind, enabling them to provide understandable rationales for their decisions during the prediction process itself. During the classification task, constraints are applied to enforce specific properties, typically with the goal of reducing overall complexity. This allows them to provide more insight into the computation of the output. Each method applies one of two constraints:

- **Information Constraint:** An information constraint limits the variability or amount of data passing through the network. This is mainly applied using the principle of an information bottleneck [34]. Imposing a bottleneck on the information allows for control over the amount of variation within the data of a network.
- **Structural Constraint:** Imposing structural constraints on the input refers to the manipulation of the input graph. The models outlined in this report belong to this category, wherein we strategically manipulate the graph to generate a sparser, more interpretable subgraph.

2.3.2. Evaluating Explanations

Explanation characteristics

In practice, an explanation is a list of scores that indicates the importance of certain entities in the network to the classification. These entities can take the form of nodes, edges, or features, which are referred to as explanations at the node-level, edge-level, or feature-level, respectively. We solely focus on structural explanations, excluding feature-level explanations. Displaying a visualization of the graph highlighting those nodes or edges creates an explanation that is interpretable for a human. However, interpretability is a fluid term and the quality of an explanation depends on its use case, e.g. should the explanation be similar to how a human would have made the decision, preferable when a model has to be trusted by humans; Or should it most accurately display the features that caused the model to make

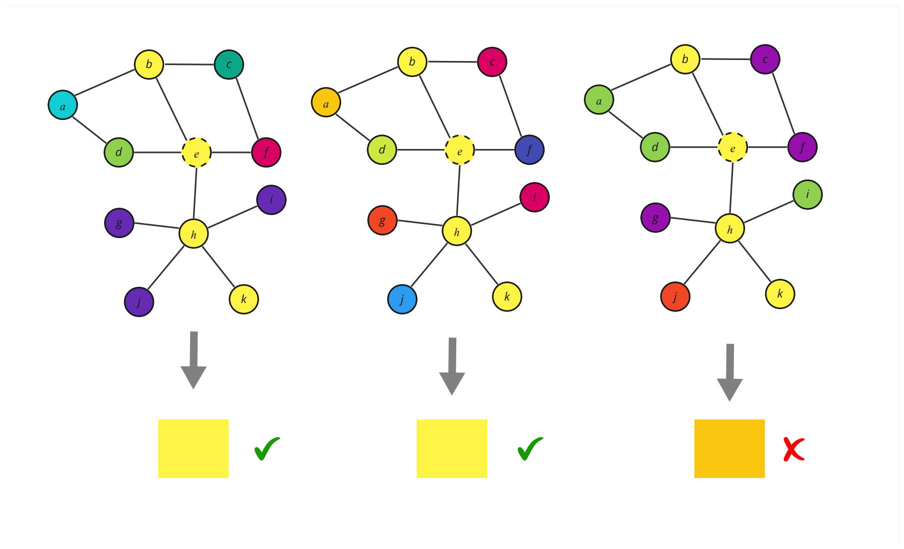


Figure 2.6: An illustration of faithfulness. If set $\{b, e, h, k\}$ forms the explanation for the classification of node e, the classification of node e should remain the same even if the features of nodes outside the explanation are changed.

the decision, preferable in cases where we solely want to understand the model. BAGEL [26], a work focussing on the evaluation of GNN explanations, identifies four diverse evaluation categories:

- **Faithfulness:** A faithful explanation accurately attributes causality to the model [12]. It should provide insights into why a particular outcome was produced, offering a clear and coherent account of the relevant features, patterns, or data points that influenced the model's output. In [26], it is argued that it is not possible to effectively capture faithfulness in a single measure. Thus, two methods are proposed.
  RDT-Fidelity [7] is proposed as a robust method to evaluate the stability of an explanation. This approach perturbs all factors outside of the explanation set and reports the effect on the classification output, as illustrated in Figure 2.6. Having a similar output for different perturbations indicates a stable explanation, producing a high fidelity score. Given explanations $\boldsymbol{p}_i \in \triangle_{n-1}$ for each node $1 \leq i \leq n$, let $\boldsymbol{M} \in [0,1]^{n\times n}$ denote a mask matrix, such that $M_{ij} = p_{ij}$. The RDT-Fidelity concerning the network Φ and the noise distribution $\mathcal{N}$, is expressed as follows:

  $$\mathcal{F}(\boldsymbol{M}) = \mathbb{E}\left[1_{\Phi(\boldsymbol{X})=\Phi(\tilde{\boldsymbol{X}}(\boldsymbol{M}))}\right], \tag{2.14}$$

  where $\boldsymbol{X} \in \mathbb{R}^{n\times n}$ represents the input, and $\tilde{\boldsymbol{X}}(\boldsymbol{M})$ is a perturbed input defined as:

  $$\tilde{\boldsymbol{X}}(\boldsymbol{M}) = \boldsymbol{M} \odot \boldsymbol{X} + (1 - \boldsymbol{M}) \odot \boldsymbol{Z}, \quad \boldsymbol{Z} \sim \mathcal{N}. \tag{2.15}$$

  As [26], we set the noise distribution as the global empirical distribution of the input features.
  The second approach computes both *comprehensiveness* and *sufficiency*. Comprehensiveness indicates whether all elements in the explanation mask are relevant and sufficiency measures whether the nodes and edges come up with the original prediction. Let $\mathcal{G}$ be the full graph and $\mathcal{G}' \subseteq \mathcal{G}$ is the explanation graph containing the important nodes/edges. Let f be the trained GAT model and $f_j(\mathcal{G})$ the prediction made by the GAT for the j^{th} class. Comprehensiveness and sufficiency are calculated by:

  $$\text{sufficiency } = f_j(\mathcal{G}) - f_j\left(\mathcal{G}'\right), \quad \text{comprehensiveness } = f_j(\mathcal{G}) - f_j\left(\mathcal{G} \setminus \mathcal{G}'\right), \tag{2.16}$$

  where $\mathcal{G} \setminus \mathcal{G}'$ indicates the set of nodes in $\mathcal{G}$ excluding the nodes also present in $\mathcal{G}'$.
  In this report, RDT-Fidelity is used to evaluate explanation interpretability. While our evaluation focuses on node-level explanations, it is important to note that the input features play a role in node selection process through the attention mechanism. The learned attention weights are derived from the input features and will determine the binary mask over the graph. Consequently, to gain a comprehensive understanding, perturbing the features during evaluation is essential.

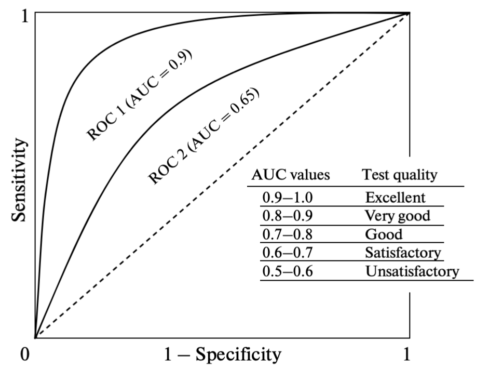


Figure 2.7: Two examples of ROC curves and their corresponding AUC scores.

Furthermore, this allows for a soft explanation mask (non-binary mask), resulting in a more nuanced indication of node importance.

- **Sparsity:** Adding all nodes and features to an explanation will result in a perfect score for the proposed measures of faithfulness. Hence, only reporting those measures is not enough. We also need to report the *sparsity* of an explanation. The sparsity can be quantified by comparing the number of bits needed to encode the explanation to the number of bits needed to encode the input. To make this comparison for a given input we determine the effective size of the explanation by calculating the Gibz-Boltmann-Shannon entropy as proposed by [26]. Specifically, we compute

$$H(p) = -\sum_{\phi \in \S} p(\phi) \log p(\phi). \tag{2.17}$$

- **Correctness:** In some cases it is important to detect any spurious correlations pickup by the model. Such correlation can increase model bias and might be injected by accident or with malicious intent. The ability to recognize these injected is measured by adding artificial edges to the nodes and measuring how many of these edges appear in the explanation after retraining the network on perturbed data. However, this metric is not evaluated in this report.
- **Plausibility:** An explanation can also be evaluated in terms of how human-like this explanation is. More easily defined as how closely the classification performed by the model is to the intuition of a human performing the same task. Although this is an interesting metric, it requires a dataset that has a human-annotated ground-truth. As these are not readily available, this measure is not evaluated in this report.

Ground-truth metrics

In specific cases a *ground-truth* explanation is available. This allows us to objectively compare our extracted rationales to the explanations provided by the dataset. To measure the performance of our explanations we report the *AUC* and *accuracy* scores.
The provided ground-truth is a binary mask whereas the extracted explanation mask ranges from 0 to 1. To calculate the accuracy a threshold of 0.5 is introduced to binarize the extracted explanation. The ratio of correct labels produces the accuracy score.
In contrast to the accuracy measure, AUC does not require a threshold. When calculating the AUC score, the sensitivity (True Positive Rate, TPR), calculated by $TPR = TP/(TP + FN)$ and the specificity (False Positive Rate, FPR), $FPR = 1 - (TN/(TN + FP))$ are compared over a range of thresholds to produce the ROC curve (Figure 2.7). The Area Under Curve (AUC) is calculated over the ROC curve to provide the score. When the AUC is approximately 0.5, the model has no discrimination capacity to distinguish between the positive and negative classes. When an AUC score of 1.0 is produced, the classes can be perfectly distinguished, indicating that the ground-truth and explanation perfectly align.

2.3.3. Datasets

To acquire ground-truth explanations the dataset either has to be annotated by humans or generated automatically.
Generated datasets are referred to as *synthetic datasets*. These datasets are artificially generated and designed to mimic real-world scenarios. They enable controlled experimentation and allow for the assessment of explanation methods under specific conditions by providing ground-truth explanations for the generated labels. For instance, the BA-Shapes dataset [44] produces a random graph with some "house"-like node structures. Nodes within such structures are specifically annotated, providing the surrounding nodes within the structure as their contextual explanation.
Human annotated datasets, referred to as *real-world datasets*, consist of non-synthetic datasets where data is collected from real-world scenarios. Evaluating explanations on these datasets provides insights into the performance of explanation methods in practical applications. To our knowledge, there are currently no existing real-world node classification datasets that offer verifiable explanations for their classifications. As such, we are not able to evaluate any ground-truth metrics for a real-world scenario.

3 Paper

MapSelect: Sparse & Interpretable Graph Attention Networks

Anonymous authors
Paper under double-blind review

Abstract

Graph Attention Networks (GATs) have shown remarkable performance in capturing complex graph structures by assigning dense attention weights over all neighbours of a node. Attention weights can act as an inherent explanation for the model output, by highlighting the most important neighbours for a given input graph. However, the dense nature of the attention layer causes a lack of focus as all edges receive some probability mass. To overcome this, we introduce *MapSelect*, a new method providing a fully differentiable sparse attention mechanism. Through user-defined constraints, MapSelect enables precise control over the attention density, acting as a continuous relaxation of the popular top-k operator. We propose two distinct variants of MapSelect: a local approach maintaining a fixed degree per node, and a global approach preserving a percentage of the full graph. Upon conducting a comprehensive evaluation of five sparse GATs in terms of sparsity, performance, and interpretability, we provide insights on the sparsity-accuracy and sparsity-interpretability trade-offs. Our results show that MapSelect outperforms robust baselines in terms of interpretability, especially in the local context, while also leading to competitive task performance on real-world datasets.

1 Introduction

Graph Attention Networks (GATs) employ the attention mechanism to weigh the importance of neighboring nodes and their features when aggregating information. This ultimately allows for a more adaptive learning for the task at hand (Veličković et al., 2018). The learned attention weights can also be inspected to gain insights into what the model considers as discriminative features towards a final decision (Ying et al., 2019; Ye & Ji, 2021; Rath et al., 2021). However, the dense nature of the attention mechanism caused by the softmax transformation, which assigns probability mass to all edges (even irrelevant ones) challenges interpretability, thereby resulting in a computation graph as dense as the input graph itself. Thresholding attention probabilities is a straightforward solution to the issue of dense attention, but it compromises the end-to-end differentiability of the network.

The importance of sparsifying GATs has been recognized by a number of works with applications to robustness, task performance, and computational efficiency (Kipf et al., 2018; Srinivasa et al., 2020; Luo et al., 2021; Shirzad et al., 2023; Ye & Ji, 2021). Particularly, the work by Rathee et al. (2021) argues that GNNs are hard to interpret and rely on sparsity to improve interpretability. Thus, despite built-in sparse methods carry potential interpretability, they are mostly focused on marginal gains in task performance. Moreover, even if there is a focus on interpretability, the ability to control the induced subgraph size, and thereby the interpretability, is often sidestepped (Rathee et al., 2021).

In this paper, motivated by the success of controllable sparse attention methods in NLP (Correia et al., 2019; Treviso & Martins, 2020; Guerreiro & Martins, 2021), we develop a novel framework named *MapSelect* that produces sparse controllable subgraphs while maintaining a high task accuracy. In particular, we use SparseMAP (Niculae et al., 2018) to create *differentiable* sparse attention masks. Differently from alternative solutions, the proposed framework can be controlled both locally and globally, through two configurations: (i) *MapSelect-L*, which produces an attention mask that maintains only the essential edges per node based on a fixed budget and (ii) *MapSelect-G*, a configuration that only maintains the most essential edges in the full graph based on a target budget. Both configurations allow for an easy control to capture the most essential edges that will provide a more focused attention mask, allowing to identify important substructures, as illustrated in Figure 1.

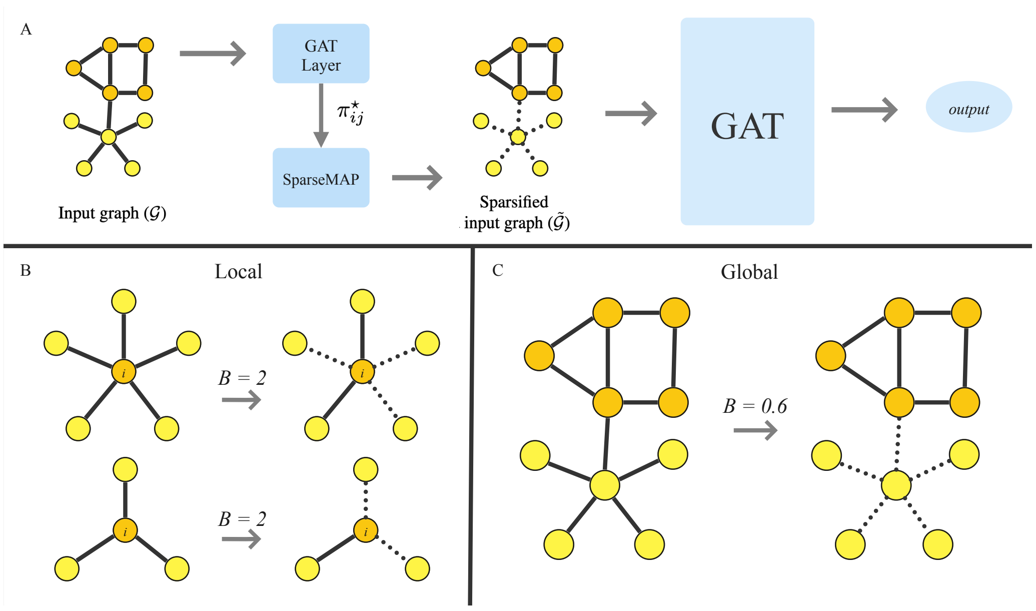


Figure 1: Overview of MapSelect. (A) The input graph is sparsified by applying SparseMAP (see §2.2) in a local or global fashion, conditioned to the information processed by a GAT layer. (B) In the local approach, MapSelect removes edges within the neighbourhood of a node, with B representing the maximum number of active connections. (C) In the global approach, MapSelect sparsifies the full graph, with B denoting the portion of active edges. In this example, in order to identify a "house" structure within a graph, MapSelect-G retains only the edges in the "house" structure.

Using five benchmark datasets, we study the effect of sparsity on both performance and interpretability, ultimately, establishing a trade-off that provides deeper insights into interpretability. We compare our method against five baselines and we also validate on a dataset with a ground-truth explanation to highlight the explanatory capability of the learned sparse attention weights. We find that MapSelect presents itself as the only method to consistently improve interpretability across all datasets, and especially on denser graphs. Overall, our contribution is twofold:[1]

1. We propose *MapSelect* to control the sparsity of graph attention layers, both locally and globally, leading to superior interpretability results compared to baselines.
2. We provide an extensive and unique evaluation of sparse GATs, examining trade-offs between sparsity-accuracy and sparsity-interpretability, and providing insights into architectural choices that influence interpretability.

2 Background

We denote a directed graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V} = \{1, ..., N\}$ and edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, where $(j, i) \in \mathcal{E}$ represents an edge from j to i, and $\mathcal{N}_i = \{j \in \mathcal{V} \mid (j, i) \in \mathcal{E}\}$ the neighborhood of node i.

2.1 Graph Attention Networks

A GAT layer computes a weighted average of vector representation of the neighbors of a node (Veličković et al., 2018). Specifically, given a set of node representations $\{\boldsymbol{h}_i^{(0)} \in \mathbb{R}^d \mid i \in \mathcal{V}\}$ as input (at layer $\ell = 0$), a GAT layer first computes attention scores for edges $(i, j) \in \mathcal{E}$ as:[2]

$$z_{ij}^{(\ell)} = \boldsymbol{a}^{(\ell)} \cdot \text{LeakyReLU}\left(\boldsymbol{W}_1^{(\ell)} \cdot \text{concat}(\boldsymbol{h}_i^{(\ell)}, \boldsymbol{h}_j^{(\ell)})\right), \tag{1}$$

[1]Our code is available at: `blind review`.

[2]We adopt the GAT variant proposed by (Brody et al., 2022), called GATv2, due to its superior expressivity.

where $\boldsymbol{a}^{(\ell)} \in \mathbb{R}^{d'}$, $\boldsymbol{W}_1^{(\ell)} \in \mathbb{R}^{d' \times 2d}$ are learnable linear transformations, and $\text{concat}(\cdot)$ denotes vector concatenation. Attention weights are then obtained by employing the softmax transformation:

$$\pi_{ij}^{(\ell)} = \text{softmax}(\boldsymbol{z}_i^{(\ell)})_j := \frac{\exp(z_{ij}^{(\ell)})}{\sum_{j' \in \mathcal{N}_i} \exp(z_{ij'}^{(\ell)})}, \tag{2}$$

where $\boldsymbol{z}_i^{(\ell)} \in \mathbb{R}^n$ represents the attention scores of node i, with $n = |\mathcal{N}_i|$. That is, softmax maps scores to probabilities $\mathbb{R}^n \to \triangle_{n-1}$, where $\triangle_{n-1} := \{\boldsymbol{\xi} \in \mathbb{R}^n \mid \boldsymbol{\xi} \geq \boldsymbol{0},\ \boldsymbol{1}^\top \boldsymbol{\xi} = 1\}$ is the $(n-1)$-probability simplex. The updated representation of node i, $\boldsymbol{h}_i^{(\ell+1)} \in \mathbb{R}^{d'}$, is determined by the weighted average of the transformed features from neighbouring nodes, potentially followed by a non-linear function $\sigma(\cdot)$:

$$\boldsymbol{h}_i^{(\ell+1)} = \sigma\left(\sum_{j \in \mathcal{N}_i} \pi_{ij}^{(\ell)} \boldsymbol{W}_2^{(\ell)} \boldsymbol{h}_j^{(\ell)}\right), \tag{3}$$

where $\boldsymbol{W}_2^{(\ell)} \in \mathbb{R}^{d' \times d}$ is a learnable linear transformation. While softmax is easy to implement and fully differentiable, its output is dense and thus all edges will have some probability mass, which may hinder interpretability. This has been investigated in (Treviso & Martins, 2020) for NLP tasks.

2.2 Sparse Attention

Previous studies have shown that incorporating sparsity into the attention mechanism results in a more compact and transparent representation of $\mathcal{G}$ (Kipf et al., 2018; Ye & Ji, 2021). In this section, we briefly review SparseMAP (Niculae et al., 2018), a technique that achieves this functionality while preserving the necessary differentiability for backpropagation. In §3, we leverage SparseMAP to introduce a new sparse attention method for GNNs.

Let $\boldsymbol{z} \in \mathbb{R}^n$ be a vector of scores given to the edges of a particular node. To improve interpretability, a possible approach is to transform $\boldsymbol{z}$ into a *sparse* probability vector $\boldsymbol{\pi}$ whose entries indicate in probability the role of an edge towards the final decision. This can be achieved by the α-entmax attention (Peters et al., 2019), a generalization of softmax, which has been to obtain sparse transformers (Correia et al., 2019). For situations where one wants to specify precisely the number of neighbors, a top-k operation can also be used, such that only the k largest entries in $\boldsymbol{\pi}$ are kept, while the others are zeroed out (Gao & Ji, 2019). However, the top-k operation is not differentiable and may inevitably introduce instabilities for training .

SparseMAP addresses this issue by casting subset selection as a relaxed structured prediction problem. Specifically, when considering n-length binary sequences with at most B non-zeros, SparseMAP is defined as:

$$\begin{aligned} \text{SparseMAP}(\boldsymbol{z}; B) := &\operatorname*{arg\,max}_{\boldsymbol{\mu} \in \mathcal{M}_\mathcal{S}} \boldsymbol{z}^\top \boldsymbol{\mu} + \frac{1}{2}\|\boldsymbol{\mu}\|_2^2 \\ \text{s.t.} \quad & \mathcal{M}_\mathcal{S} := \big\{ \sum_i \pi_i \boldsymbol{y}_i : \boldsymbol{\pi} \in \triangle_{|\mathcal{S}|-1}, \boldsymbol{y}_i \in \mathcal{S} \big\} \\ & \mathcal{S} := \{\boldsymbol{y} \in \{0,1\}^n \mid \|\boldsymbol{y}\|_1 \leq B\}, \end{aligned} \tag{4}$$

where the cost function aims at finding a vector $\boldsymbol{\mu}$ that is aligned with the scores $\boldsymbol{z}$ but with a quadratic regularizer to ensure smoothness. The solution is confined to the marginal polytope $\mathcal{M}_\mathcal{S}$ that imposes solutions to be in the space of bit vectors $[0,1]^n$ with at most B non-zeros. Notably, the vertices of $\mathcal{M}_\mathcal{S}$ represent binary solutions and in such cases we obtain $\boldsymbol{u} \in \{0,1\}^n$, whereas its edges represent a sparse convex combination of binary vectors, and thus $\boldsymbol{u} \in [0,1]^n$. Finally, the faces of this polytope lead to fully dense solutions. Because of the quadratic regularization term, SparseMAP promotes sparse vectors $\boldsymbol{u} \in [0,1]^n$ that lie on the boundary of the marginal polytope (vertices, edges, or other low-dimensional faces). For more information on SparseMAP, we refer the reader to (Niculae et al., 2018; Niculae & Martins, 2020).

Therefore, given the vector scores $\boldsymbol{z}_i \in \mathbb{R}^n$ of node $i \in \mathcal{V}$, SparseMAP will produce a vector $\boldsymbol{\mu}_i \in [0,1]^n$ as output, such that edges with $\mu_{ij} = 0$ can be ignored during the forward pass. While the optimization problem described in Equation 4 does not have a closed-form solution, both the forward and backward passes can be solved with an active set method that exhibits exact finite convergence and yields the optimal sparsity pattern (Nocedal & Wright, 1999). Contrarily to stochastic approaches, such as the reparameterization trick used in NeuralSparse (Zheng et al., 2020) and SGAT (Ye & Ji, 2021), SparseMAP is deterministic and end-to-end differentiable, and thus easier to optimize. Next, we present MapSelect, a new sparse method for GNNs that leverages SparseMAP.

3 MapSelect

We introduce two methods that leverage sparsity to design more interpretable GNNs by acting on different levels of the computation graph. The first method, **MapSelect-L**, keeps a sparse subset of local connections for each node, while the second approach, **MapSelect-G**, promotes sparsity on the full computation graph; see Figure 1 for an overview. For both approaches, we start with a GAT layer that takes the input graph representation and produces the attention scores $z^\star_{ij}$ and the attention weights $\pi^\star_{ij}$ for each edge $(i, j) \in \mathcal{E}$, as described in §2.1. EWe pass the attention weights (MapSelect-L) or the attention scores (MapSelect-G) to SparseMAP (cf. Equation 4) and obtain a sparse distribution as output, which we leverage to obtain a sparse input graph $\tilde{\mathcal{G}}$ that is used in subsequent GAT layers. Next, we detail each variant of MapSelect.

MapSelect-L. In this approach, we fix a local budget B per node, such that each node keeps at most B active connections. Formally, for each node i, let $\boldsymbol{\pi}^\star_i \in \triangle_{n-1}$ be the attention weights obtained with the first GAT layer, where $n = |\mathcal{N}(i)|$ and let $\mathrm{SparseMAP}(\cdot; B)$ denote the SparseMAP with a budget constraint B. In MapSelect-L, we apply SparseMAP on $\boldsymbol{\pi}^\star_i$ with a budget constraint to get a sparse mask $\boldsymbol{\mu}_i \in [0, 1]^n$:

$$\boldsymbol{\mu}_i = \mathrm{SparseMAP}(\boldsymbol{\pi}^\star_i / t; B), \tag{5}$$

where $t \in \mathbb{R}$ is a temperature hyperparameter. Next, we use $\boldsymbol{\mu}_i$ to re-scale the attention weights of each subsequent ℓ-th GAT layer as:[3]

$$\tilde{\boldsymbol{\pi}}^{(\ell)}_i = \frac{\boldsymbol{\mu}_i \odot \boldsymbol{\pi}^{(\ell)}_i}{\boldsymbol{\mu}_i^\top \boldsymbol{\pi}^{(\ell)}_i}, \quad \boldsymbol{h}^{(\ell+1)}_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \tilde{\pi}^{(\ell)}_{ij} \boldsymbol{W}^{(\ell)}_2 \boldsymbol{h}^{(\ell)}_j\right), \tag{6}$$

where $\odot$ represents the element-wise multiplication, and $\boldsymbol{h}^{(0)}_i$ the feature vector fed to the network (see Figure 1). This procedure effectively deactivates the contribution of neighbouring nodes $j \in \mathcal{N}_i$ when $\mu_{ij} = 0$. In other words, we condition SparseMAP on information processed by a GAT layer and then use its output to sparsify the input graph by adjusting subsequent attention layers. Notably, as the temperature $t \to 0$, SparseMAP becomes the top-B operator and $\tilde{\boldsymbol{\pi}}_i$ becomes a re-normalized vector of probabilities with the top-B highest original probabilities. Therefore, our proposed framework can also be seen as a continuous relaxation of the usual truncation approach.[4]

Remark 1. The current approach imposes the budget B as the maximum number of edges allowed per node. In some cases, where statistical properties of the input graph (such as degree distribution or centrality metrics) may be relevant this strategy can be changed by using a relative budget ($B\%$) on the available number of edges. In our experiments, we tested both approaches but have not noticed a significant impact on performance. ■

MapSelect-G. MapSelect-G evokes SparseMAP over all edges globally, disregarding specific neighbourhoods. Here, we set the budget B as a percentage of the number of edges as:

$$\bar{\boldsymbol{z}}^\star = \text{concat-and-pad}(\boldsymbol{z}^\star_1, ..., \boldsymbol{z}^\star_N) \tag{7}$$

$$\bar{\boldsymbol{\mu}} = \mathrm{SparseMAP}(\bar{\boldsymbol{z}}^\star / t; B), \tag{8}$$

where $\bar{z}^\star \in \mathbb{R}^{N^2}$ is the concatenation of all attention scores $\boldsymbol{z}^\star_i$ from the 1st GAT layer for all $1 \leq i \leq N$ nodes, padding (i, j) positions with $-\infty$ when $(i, j) \notin \mathcal{E}$. Here, we denote $\boldsymbol{\mu}_i \in [0, 1]^N$ as the binary vector given to node i, indexed as the i-th contiguous chunk of size N in $\bar{\boldsymbol{\mu}} \in [0, 1]^{N^2}$. As in MapSelect-L, we use $\boldsymbol{\mu}_i$ to re-scale the attention weights of node i in subsequent GAT layers (see Equation 6), keeping self-loops by setting $\mu_{ii} = 1$. Therefore, edges $(i, j) \in \mathcal{E}$ will be deactivated whenever $\mu_{ij} = 0$, and as a result, they will not contribute towards the final output.

[3]We ensure $\boldsymbol{\mu}_i \neq \mathbf{0}$ by keeping self-loops (i.e., $\mu_{ii} = 1$ always).

[4]We employ a temperature parameter of 10^{-1} and 10^{-3} in training and test time, respectively.

Connections with related approaches. Both MapSelect-L and MapSelect-G resemble techniques that sparsify the input graph and then use it in a classification task, such as NeuralSparse (Zheng et al., 2020), SGAT (Ye & Ji, 2021), and DropEdge (Rong et al., 2020). However, MapSelect differs by: (i) leveraging SparseMAP to sparsify the input graph, effectively keeping the classification problem end-to-end-differentiable; and (ii) applying the resulting mask to the attention mechanism in subsequent GAT layers instead of masking irrelevant connections directly in the adjacency matrix. More specifically, MapSelect-L is similar to NeuralSparse and the traditional top-k attention, as the selection of relevant edges occurs in the neighbourhood of each node in the computation graph and the selection budget is set to a pre-defined fixed number of edges. MapSelect-G is close in spirit to SGAT and DropEdge, as the decision to deactivate irrelevant edges is carried globally over the entire input graph. Finally, the way MapSelect conditions on the initial GAT layer mirrors the design seen in models termed ”rationalizers” within the NLP literature (Lei et al., 2016; Bastings et al., 2019; Guerreiro & Martins, 2021). Much like MapSelect, these models aim to provide faithful explanations by conditioning the selection of input elements (e.g., words) on an encoder module, and subsequently making a final decision solely on the basis of these selected items.

4 EVALUATION

We compare the proposed methods to five baselines that focus on producing a sparse subset of the input graph. We perform experiments on five real-world datasets, and on one synthetic dataset containing ground truth explanations. The detailed model configurations and the dataset information can be found in §B.1 and §B.3, respectively.

4.1 BASELINES

We assess the proposed approaches by comparing them to the following alternatives. A summary of the characteristics of each method is presented in Table 1.

Table 1: Characteristics of each baseline method.

Method	Sparsity Level	Sparsity Control	End-to-end Differentiable
Top-k	Local	✓	✗
Entmax	Local	✗	✓
NeuralSparse	Local	✓	✗
MapSelect-L	Local	✓	✓
SGAT	Global	✗	✗
DropEdge	Global	✓	✗
MapSelect-G	Global	✓	✓

Top-k. We apply a top-k operation on the softmax attention weights of a standard two-layer GAT in a local fashion. We control for sparsity by varying k at test time.

Entmax. This approach replaces the standard softmax function found in GATs by the α-entmax transformation (Peters et al., 2019), detailed in §A. We control the propensity to sparsity by varying α. Both Top-k and Entmax produce sparse attention probabilities directly rather than implementing a separate attention layer that masks the input graph.

NeuralSparse. (Zheng et al., 2020) NeuralSparse utilizes Gumbel-Softmax (Jang et al., 2017) to sample (local) sparse subgraphs consisting of k neighbours. We control sparsity by setting k as the maximum number of edges per node.

SGAT. (Ye & Ji, 2021) SGAT encourages sparse solutions by adding an ℓ_0-penalty term to the loss function, penalizing non-zero attention weights, resulting in global sparsification. SGAT resorts to the hard concrete estimator for model optimization (Louizos et al., 2018). We control sparsity by adjusting the weight given to the ℓ_0 penalty empirically.

DropEdge. (Rong et al., 2020) DropEdge randomly drops edges from the input graph, thus acting in a global fashion. We consider this method as a baseline by controlling the portion of dropped edges and maintaining the sparsified graph at test time.

4.2 EXPERIMENTAL SETUP

MapSelect. For both variants of MapSelect, a single GAT layer is employed to derive the set of attention weights $\pi_i^\star$, which are used to form the sparse mask. Following this, two GAT layers are employed to classify the input using the masked attention.[5] We control the sparsity of MapSelect

[5]To ensure a fair comparison, we use two GAT layers for classification in all methods, including MapSelect, regardless of whether the input is a full or sparsified graph.

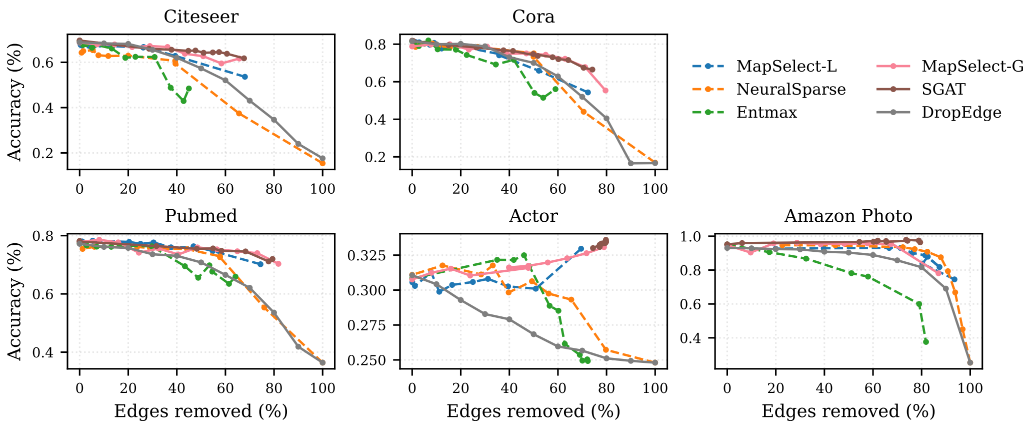


Figure 2: The impact of sparsity on the model performance.

by adjusting the SparseMAP's budget constraint B. In the local approach, B can alternatively be configured to retain a specific percentage of connections. However, for the sake of uniformity with NeuralSparse and due to similar performance, we only assess the fixed configuration.

Metrics. We evaluate interpretability with the fidelity metric proposed by ZORRO (Funke et al., 2023). More details on fidelity can be found in §B.2. Since the synthetic dataset provides binary vectors as ground truth explanations, for this dataset we also compare our explanations with respect to the ground truth in terms of AUC, which automatically accounts for multiple binarization thresholds. Furthermore, an evaluation of the explanation entropy, as proposed by BAGEL (Rathee et al., 2022), is presented in §D.1.

Explanation extraction. We employ two distinct strategies for extracting explanations. For real-world datasets, we obtain *node-level explanations* by propagating an identity matrix over the computation graph. We detail this strategy in §C. For the synthetic dataset, we follow the approach proposed by (Ying et al., 2019), which produces *edge-level explanations* by averaging the attention scores of all layers.

4.3 Experiments

We hypothesize that as we progressively remove edges from the computation graph, the performance will decline. In addition, we anticipate that a classification based on fewer edges will be more interpretable To investigate these effects independently and identify a balance between them, we pose the following research questions:

RQ1. *What is the role of sparsity on model performance?*

RQ2. *What is the role of sparsification on model interpretability?*

RQ3. *What is the interpretability-performance trade-off?*

4.3.1 Role of sparsity on task performance

In Figure 2, we present results for all methods on real-world datasets, with graphs sorted from the least to the most dense. Among the local methods, MapSelect-L consistently outperforms NeuralSparse and top-k. Notably, unlike its counterparts, MapSelect-L is not tied to a specified budget, giving us the flexibility to select fewer edges than the targeted allocation. MapSelect-L shows also a more stable convergence as more edges are discarded than the other local approaches.

Looking at the global methods, both MapSelect-G and SGAT surpass DropEdge. In the less dense graphs, SGAT and MapSelect-G achieve similar results, while in more dense datasets SGAT outperforms MapSelect-G. However, SGAT faces challenges in maintaining sparsity control in dense graphs, as it quickly deviates from $\sim 10\%$ to $\sim 60\%$ sparsity. In contrast, MapSelect-G provides a tight control over sparsity, respecting the desired budget pre-established before training. Regarding the impact of sparsity on accuracy, we note that both SGAT and MapSelect-G can maintain or even

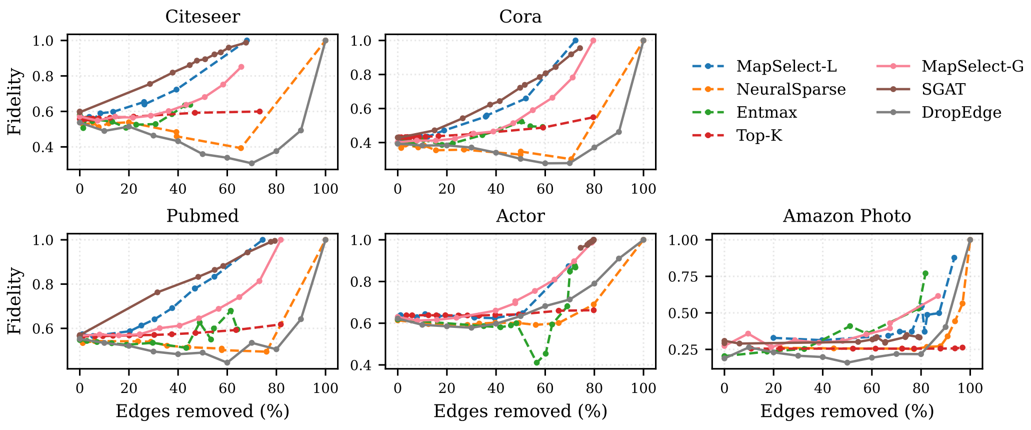


Figure 3: Impact of sparsity on interpretability.

improve accuracy as sparsity increases on the Actor dataset, suggesting that this dataset might contain a considerable number of irrelevant edges. This is expected because global approaches impose fewer limitations on which edges to remove while during training.

Overall, we observe that global methods typically outperform local approaches when the primary focus is on task performance, likely because global approaches impose fewer limitations on edge removal. In a case where it is more beneficial to maintain all edges for one node and remove all edges for a different one, a global approach should be considered. Interestingly though, the local approaches achieve a similar performance on the more dense datasets and sometimes they outperform the global approaches in the most sparsified settings. In addressing RQ1, we find that sparsity presents a nuanced trade-off in task performance. While extreme sparsity can indeed lead to decreased performance, a moderate degree of sparsity, around 40%, results in a minimal performance drop, often less than 5% across all datasets, especially on denser ones.

4.3.2 Role of sparsity on interpretability

Towards answering RQ2, we evaluate the tradeoff between sparsity and interpretability on real-world datasets first, and then move to the synthetic dataset with ground-truth explanations.

Real-world datasets. In Figure 3, we present the impact of sparsity on fidelity. Intuitively, a high fidelity implies that the explanation is more faithful and is more robust to perturbations. Among the local methods, the results vary as we increase the sparsity rate. Initially, top-k has a better fidelity than MapSelect-L, however, as we remove more edges, MapSelect-L consistently outperforms other methods. The early success of Entmax and top-k might be due to a better attention distribution. This is supported by the lower entropy of Entmax, explored in §D.1. Regarding the global approaches, MapSelect-G outperforms DropEdge and SGAT is the best performing among the global methods.

In contrast to our findings in terms of the sparsity-accuracy tradeoff, global methods do not always lead to a better interpretability than local the approaches. In the context of MapSelect, we see a trend towards preferring local sparsification, which achieves a results competitive to SGAT. Lastly, we remark that graph density significantly impacts the interpretability. For example, while SGAT has the overall best interpretability results, MapSelect-L outperforms it in the Amazon Photo dataset.

Synthetic dataset. To investigate whether an extracted rationale agrees with the ground truth explanation, we evaluate the models on the BA-Shapes dataset. We show the trade-off between sparsity and AUC in Figure 4. The task accuracy of each method can be found in §D.2.

For local methods, we observe a standout performance from MapSelect-L. As more edges are removed, its AUC score increases. However, after removing more than 60% of edges, the score drops. This indicates that MapSelect-L provides better explanations with a moderate sparsity rate. NeuralSparse shows a more significant increase in AUC but starts with a much lower initial score. Top-k performs as expected, producing less faithful explanations as more edges are removed.

Turning to global methods, SGAT outperforms other approaches, mirroring the trend seen with MapSelect-L. The trajectory of MapSelect-G starts with high AUC scores, and then we get lower scores as sparsity increases. The lower performance of MapSelect-L compared with MapSelect-G can be attributed to the nature of the BA-Shapes dataset, which emphasizes the discovery of small structures within a vast graph, deeming all other edges irrelevant. These irrelevant edges are retained in the local approach, as it keeps only a small absolute number of edges per node.

From the increased trend in the AUC scores, we conclude that attention-based methods can be explored to extract plausible explanations. Notably, both MapSelect-G and SGAT outperform the AUC scores presented by the attention, gradient, and GNNExplainer baselines in Luo et al. (2020), with SGAT even outperforming the proposed PGExplainer itself. We show an example of an explanation extracted with MapSelect-G for BA-Shapes in Figure 1C.

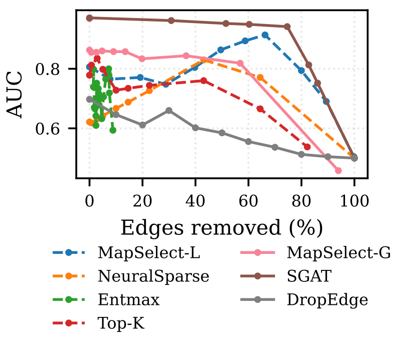


Figure 4: The impact of graph sparsification on the similarity of the extracted rationale to the ground truth explanation (in terms of AUC) on the BA-Shapes dataset.

4.3.3 Performance-interpretability Trade-off

As seen in Figure 2 and Figure 3, the lowest accuracy and the best fidelity scores are reported when most edges are removed. Since a consistent explanation of a wrong classification may be irrelevant, we investigate the effect of interpretability in accuracy directly in Figure 5. For clarity, we removed the baselines that did not show sufficient improvement in interpretability.

First, we can see that MapSelect-L is the only local method offering an appropriate and consistent trade-off between accuracy and fidelity. Second, we see that MapSelect-G is more suitable for denser datasets (e.g., Amazon Photo), where its ability to control the sparsity allows improving the fidelity score by up to a factor of two while retaining the accuracy. Contrarily, SGAT works best in sparser datasets but in denser ones it strugles to enhance the fidelity due to limited control over edge removal. Both MapSelect methods consistently demonstrate their ability to yield more interpretable networks by robustly removing edges in all scenarios. The preference for either the local or global approach appears to strongly hinge on the task as well as the dataset. Overall, these analyses show that studying the trajectory of task accuracy and interpretability score as we change sparsity reveals a more profound understanding of the capabilities of each method.

5 Related Work

Graph sparsification can be applied for a variety of goals, such as robustness, mitigating oversmoothing, removing noise, decreasing computation times or improving interpretability. Here, we focus on existing works that apply sparsity to improve interpretability.

Post-hoc approaches. The majority of methods targeting the enhancement of GNN interpretability adopt a *post-hoc* approach, pinpointing relevant nodes and features for decisions made by a trained network. Significant contributions in this category include GNNExplainer (Ying et al., 2019), PGExplainer (Luo et al., 2020), XGNN (Yuan et al., 2020), GraphMask (Schlichtkrull et al., 2021), and Zorro (Funke et al., 2023). Although post-hoc methods can be readily applied to any black-box model, they overlook the intrinsic explainable elements of the model, potentially compromising their faithfulness (Rudin, 2019; Kakkad et al., 2023).

Local self-interpretable methods. A second perspective of interpretability is given by *self-interpretable* approaches. Contrasting with post-hoc methods, these are integrated directly within the model's architecture. Within this view, the subgraph generated during forward propagation can be considered a faithful explanation for a particular decision. For instance, NeuralSparse (Zheng et al., 2020) learns a k-neighbor subgraph for each node by sampling an adjacency from a Gumbel-Softmax distribution and employs the reparametrization trick to address non-differentiability.

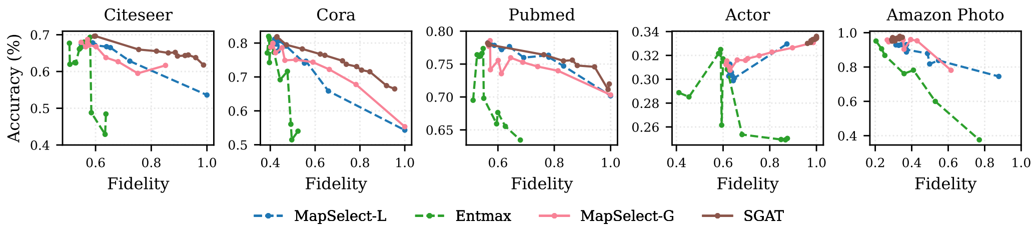


Figure 5: The trade-off between the accuracy and the fidelity score.

SEGNN (Meng et al., 2022) constructs an explanation subgraph by grouping k nodes with similar structure and features, learning the grouping process by adding a contrastive penalty to the loss function. In contrast, MapSelect can also act locally within a neighborhood (MapSelect-L), while still being deterministic and end-to-end differentiable without requiring a multi-task objective.

Global self-interpretable methods. Other *self-interpretable* methods seek to induce sparsity globally, without restricting this process to a specific neighborhood. This goal is shared by PTD-Net (Luo et al., 2021), KEdge (Rathee et al., 2021), SGAT Ye & Ji (2021), and others (Feng et al., 2022; Zhang et al., 2022; Miao et al., 2022). Analogous to NeuralSparse, PTDNet samples a subgraph that is used for classification, but its sparsity is imposed via a loss penalty, complicating its controllability. KEdge (Rathee et al., 2021) produces a subgraph by sampling binary masks from a HardKuma distribution over the adjacency matrix. Meanwhile, SGAT (Ye & Ji, 2021) prunes edges through attention weights, but requires sampling from a Hard-Concrete distribution. Both KEdge and SGAT resort to the reparameterization trick for optimization. MapSelect-G aligns with SGAT in its methodology, but with SparseMAP, the selection is entirely differentiable and flexible, allowing users to define a specific sparsity budget. Finally, we note that differently to MapSelect, SGAT, PTD-NET, and NeuralSparse do not primarily concentrate on improving interpretability; rather, it emerges as a by-product of their built-in sparse approaches.

Connections to rationalizers in NLP. As mentioned in §3, MapSelect aligns with the objectives of rationalizers, colloquially termed *mask-then-predict* techniques, which are prevalent in NLP for extracting faithful explanations (Jain et al., 2020; Jacovi & Goldberg, 2020). Classical examples include rationalizers that sample masks from a Bernoulli (Lei et al., 2016) or HardKuma distribution (Bastings et al., 2019). Addressing training instabilities triggered by stochastic estimators, Treviso & Martins (2020) suggests leveraging the α-entmax transformation (Peters et al., 2019) for the selection mechanism. Guerreiro & Martins (2021) introduced SPECTRA, a method providing differentiability and control over sparsity through SparseMAP (Niculae & Martins, 2020), exhibiting superiority over the aforementioned stochastic alternatives. In this work, we assess α-entmax attention as baseline in §4. While both MapSelect and SPECTRA incorporate SparseMAP, MapSelect is specifically tailored for graph structures, allowing for both local and global applications.

6 Conclusion

We presented MapSelect, a method to learn sparse and interpretable attention scores in graph neural networks. MapSelect relies on SparseMAP, conventionally used in NLP (Guerreiro & Martins, 2021), to prune the attention scores both in a locally and globally controlled manner. The local approach, MapSelect-L, is more beneficial when we deal with node-centric tasks and want to enhance the sparsity at the surroundings of each node. The global approach, MapSelect-G, is more beneficial when we deal with graph-centric tasks and focus on the whole graph sparsity without any local constraints. Upon studying different trade-offs between sparsity, task performance, and interpretability, MapSelect-L achieved consistently the best performance w.r.t. different state-of-the-art alternatives in five datasets. Instead, MapSelect-G showed that it is more appropriate than alternative sparse solutions on denser graphs, where its stronger ability to control sparsity proved beneficial. By controlling the sparsity of the graph, the proposed approaches carry the potential advantage of overcoming over-smoothing Rathee et al. (2021) and over-squashing (Alon & Yahav, 2021). Such a task could be achieved by introducing different constraints into the MapSelect such as maximum spanning tree constraints and will be studied in future work.

REFERENCES

Uri Alon and Eran Yahav. On the Bottleneck of Graph Neural Networks and its Practical Implications. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=i80OPhOCVH2.

Jasmijn Bastings, Wilker Aziz, and Ivan Titov. Interpretable Neural Predictions with Differentiable Binary Variables. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2963–2977, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1284. URL https://aclanthology.org/P19-1284.

Shaked Brody, Uri Alon, and Eran Yahav. How Attentive are Graph Attention Networks? . In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=F72ximsx7C1.

Gonçalo M. Correia, Vlad Niculae, and André F. T. Martins. Adaptively Sparse Transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2174–2184, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1223. URL https://aclanthology.org/D19-1223.

Aosong Feng, Chenyu You, Shiqiang Wang, and Leandros Tassiulas. KerGNNs: Interpretable Graph Neural Networks with Graph Kernels. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(6):6614–6622, Jun. 2022. doi: 10.1609/aaai.v36i6.20615. URL https://ojs.aaai.org/index.php/AAAI/article/view/20615.

Matthias Fey and Jan E. Lenssen. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Kimon Fountoulakis, Amit Levi, Shenghao Yang, Aseem Baranwal, and Aukosh Jagannath. Graph Attention Retrospective. *Journal of Machine Learning Research*, 24(246):1–52, 2023. URL http://jmlr.org/papers/v24/22-125.html.

T. Funke, M. Khosla, M. Rathee, and A. Anand. Zorro: Valid, Sparse, and Stable Explanations in Graph Neural Networks. *IEEE Transactions on Knowledge & Data Engineering*, 35(08):8687–8698, aug 2023. ISSN 1558-2191. doi: 10.1109/TKDE.2022.3201170. URL https://doi.ieeecomputersociety.org/10.1109/TKDE.2022.3201170.

Hongyang Gao and Shuiwang Ji. Graph U-Nets. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2083–2092. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/gao19a.html.

Nuno M. Guerreiro and André F. T. Martins. SPECTRA: Sparse Structured Text Rationalization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 6534–6550, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.525. URL https://aclanthology.org/2021.emnlp-main.525.

Alon Jacovi and Yoav Goldberg. Towards Faithfully Interpretable NLP Systems: How Should We Define and Evaluate Faithfulness? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4198–4205, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.386. URL https://aclanthology.org/2020.acl-main.386.

Sarthak Jain, Sarah Wiegreffe, Yuval Pinter, and Byron C. Wallace. Learning to Faithfully Rationalize by Construction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4459–4473, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.409. URL https://aclanthology.org/2020.acl-main.409.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical Reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=rkE3y85ee.

Jaykumar Kakkad, Jaspal Jannu, Kartik Sharma, Charu Aggarwal, and Sourav Medya. A Survey on Explainability of Graph Neural Networks. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 2023. URL `http://sites.computer.org/debull/A23june/p35.pdf`.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural Relational Inference for Interacting Systems. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2688–2697. PMLR, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/kipf18a.html`.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. Rationalizing Neural Predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 107–117, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1011. URL `https://aclanthology.org/D16-1011`.

Christos Louizos, Max Welling, and Diederik P. Kingma. Learning Sparse Neural Networks through L0 Regularization. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=H1Y8hhg0b`.

Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized Explainer for Graph Neural Network. In *Advances in Neural Information Processing Systems*, volume 33, pp. 19620–19631. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/e37b08dd3015330dcbb5d6663667b8b8-Paper.pdf`.

Dongsheng Luo, Wei Cheng, Wenchao Yu, Bo Zong, Jingchao Ni, Haifeng Chen, and Xiang Zhang. Learning to Drop: Robust Graph Neural Network via Topological Denoising. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21, pp. 779–787, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450382977. doi: 10.1145/3437963.3441734. URL `https://doi.org/10.1145/3437963.3441734`.

Andre Martins and Ramon Astudillo. From Softmax to Sparsemax: A Sparse Model of Attention and Multi-Label Classification. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1614–1623, New York, New York, USA, 20–22 Jun 2016. PMLR. URL `https://proceedings.mlr.press/v48/martins16.html`.

Lin Meng, Haoran Yang, and Jiawei Zhang. Stage Evolving Graph Neural Network based Dynamic Recommendation with Life Cycles. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pp. 01–08, 2022. doi: 10.1109/IJCNN55064.2022.9892418.

Siqi Miao, Mia Liu, and Pan Li. Interpretable and Generalizable Graph Learning via Stochastic Attention Mechanism. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 15524–15543. PMLR, 17–23 Jul 2022. URL `https://proceedings.mlr.press/v162/miao22a.html`.

Vlad Niculae and Andre Martins. LP-SparseMAP: Differentiable Relaxed Optimization for Sparse Structured Prediction. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7348–7359. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/niculae20a.html`.

Vlad Niculae, Andre Martins, Mathieu Blondel, and Claire Cardie. SparseMAP: Differentiable Sparse Structured Inference. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3799–3808. PMLR, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/niculae18a.html`.

Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-GCN: Geometric Graph Convolutional Networks. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=S1e2agrFvS`.

Ben Peters, Vlad Niculae, and André F. T. Martins. Sparse Sequence-to-Sequence Models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1504–1519, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1146. URL `https://aclanthology.org/P19-1146`.

Bhavtosh Rath, Xavier Morales, and Jaideep Srivastava. SCARLET: Explainable Attention Based Graph Neural Network for Fake News Spreader Prediction. In *Advances in Knowledge Discovery and Data Mining*, pp. 714–727, Cham, 2021. Springer International Publishing. ISBN 978-3-030-75762-5.

Mandeep Rathee, Zijian Zhang, Thorben Funke, Megha Khosla, and Avishek Anand. Learnt sparsification for interpretable graph neural networks. *arXiv preprint arXiv:2106.12920*, 2021. URL `https://arxiv.org/abs/2106.12920`.

Mandeep Rathee, Thorben Funke, Avishek Anand, and Megha Khosla. BAGEL: A Benchmark for Assessing Graph Neural Network Explanations. *arXiv preprint arXiv:2206.13983*, 2022. URL `https://arxiv.org/abs/2206.13983`.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=Hkx1qkrKPr`.

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215, 2019. URL `https://www.nature.com/articles/s42256-019-0048-x`.

Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. Interpreting Graph Neural Networks for NLP With Differentiable Edge Masking. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=WznmQa42ZAx`.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of Graph Neural Network Evaluation. *Relational Representation Learning Workshop, NeurIPS 2018*, 2018.

Hamed Shirzad, Ameya Velingker, Balaji Venkatachalam, Danica J. Sutherland, and Ali Kemal Sinop. Exphormer: Sparse Transformers for Graphs. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 31613–31632. PMLR, 23–29 Jul 2023. URL `https://proceedings.mlr.press/v202/shirzad23a.html`.

Rakshith S Srinivasa, Cao Xiao, Lucas Glass, Justin Romberg, and Jimeng Sun. Fast graph attention networks using effective resistance based graph sparsification. *arXiv preprint arXiv:2006.08796*, 2020. URL `https://arxiv.org/abs/2006.08796`.

Marcos Treviso and André F. T. Martins. The Explanation Game: Towards Prediction Explainability through Sparse Communication. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pp. 107–118, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.blackboxnlp-1.10. URL `https://aclanthology.org/2020.blackboxnlp-1.10`.

Constantino Tsallis. Possible generalization of Boltzmann-Gibbs statistics. *Journal of Statistical Physics*, 1988.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rJXMpikCZ`.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting Semi-Supervised Learning with Graph Embeddings. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 40–48, New York, New York, USA, 20–22 Jun 2016. PMLR. URL `https://proceedings.mlr.press/v48/yanga16.html`.

Yang Ye and Shihao Ji. Sparse graph attention networks. *IEEE Transactions on Knowledge and Data Engineering*, 2021. URL `https://ieeexplore.ieee.org/document/9399811`.

Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. GNNExplainer: Generating Explanations for Graph Neural Networks. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper_files/paper/2019/file/d80b7040b773199015de6d3b4293c8ff-Paper.pdf`.

Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. XGNN: Towards Model-Level Explanations of Graph Neural Networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD ’20, pp. 430–438, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379984. doi: 10.1145/3394486.3403085. URL `https://doi.org/10.1145/3394486.3403085`.

Zaixi Zhang, Qi Liu, Hao Wang, Chengqiang Lu, and Cheekong Lee. ProtGNN: Towards Self-Explaining Graph Neural Networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(8):9127–9135, Jun. 2022. doi: 10.1609/aaai.v36i8.20898. URL `https://ojs.aaai.org/index.php/AAAI/article/view/20898`.

Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, and Wei Wang. Robust Graph Representation Learning via Neural Sparsification. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 11458–11468. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/zheng20d.html`.

A α-ENTMAX

The α-entmax transformation (Peters et al., 2019) is a natural way to obtain a sparse attention distribution from a given vector of scores, $\boldsymbol{z} \in \mathbb{R}^n$. It is defined as the regularized argmax problem:

$$\alpha\text{-entmax}(\boldsymbol{z}) := \underset{\boldsymbol{\pi} \in \triangle_{n-1}}{\arg\max}\, \boldsymbol{z}^\top \boldsymbol{\pi} + H_\alpha(\boldsymbol{\pi}), \tag{9}$$

where H_α is a generalization of the Shannon and Gini entropies proposed by (Tsallis, 1988), parameterized by a scalar $\alpha \geq 0$:

$$H_\alpha(\boldsymbol{\pi}) := \begin{cases} \frac{1}{\alpha(\alpha-1)} \sum_j (\pi_j - \pi_j^\alpha), & \alpha \neq 1 \\ -\sum_j \pi_j \log \pi_j, & \alpha = 1. \end{cases} \tag{10}$$

Given the attention scores $\boldsymbol{z}_i \in \mathbb{R}^n$ of node i, the attention weights of α-entmax can be computed in a thresholded form:[6]

$$\pi_{ij} = \alpha\text{-entmax}(\boldsymbol{z}_i)_j = [(\alpha - 1) z_{ij} - \tau(\boldsymbol{z}_i)]_+^{1/\alpha-1}, \tag{11}$$

where $[\cdot]_+$ is the ReLU function, and $\tau : \mathbb{R}^n \to \mathbb{R}$ is a normalizing function to ensure $\sum_j \pi_{ij} = 1$. Scalar α determines the propensity of sparsity: with $\alpha = 1$, α-entmax simplifies to the softmax function, whereas for $\alpha > 1$, it returns sparse solutions. As α increases, the resulting probability distribution becomes more sparse. For $\alpha = 2$, the transformation recovers sparsemax (Martins & Astudillo, 2016), defined as the Euclidean projection of $\boldsymbol{z}_i$ onto the probability simplex. We refer to Peters et al. (2019) on how to compute $\tau(\cdot)$ efficiently in $O(n \log n)$.

We use α-entmax in §4 as baseline. Remarkably, in a GAT setup, edges with a score $z_{ij} \leq \tau(\boldsymbol{z}_i)/\alpha-1$ will receive zero probability (i.e., $\pi_{ij} = 0$), and therefore can be excluded from the computation

[6]We drop the dependence on the layer ℓ for ease of exposition.

graph. Since α-entmax promotes solutions that hit the boundary of the simplex (discouraging uniform distributions), it can mitigate the lack of expressiveness present in softmax-based GATs (Brody et al., 2022; Fountoulakis et al., 2023). Still, α-entmax is restricted to produce solutions in the probability simplex, which may limit its applicability towards sparsifying the computation graph globally.

B Experimental Setup

B.1 Training

For all models, we employ the cross-entropy loss for training and optimize the loss with Adam (Kingma & Ba, 2015). For MapSelect-L, we found that feeding SparseMAP with attention weights rather than raw attention scores works better in practice. Similarly, for MapSelect-G, we found that applying an exponential operation before passing scores to SparseMAP improves stability. We report average numbers of five distinct random seeds. We used a single machine equipped with a GeForce RTX 2080 Ti (11GB) GPU. We summarize relevant training hyperparameters in Table 2.

Table 2: Training hyperparameters.

Hyperparam.	**CiteSeer**	**Cora**	**PubMed**	**Actor**	**Amazon Photos**	**BA-Shapes**
Hidden size	8	8	8	8	8	20
Dropout	0.6	0.6	0.6	0.3	0.6	0.1
Learning rate	0.01	0.01	0.01	0.01	0.01	0.01
Weight decay	0.0005	0.0005	0.01	0.0005	0.0005	0.001

Concerning the model architecture, all approaches conduct their classification using two GAT layers. Methods that incorporate a masking layer, such as MapSelect and NeuralSparse, include an additional layer dedicated to learning a graph mask directly from the input. To ensure a balanced comparison across all methods, this masking layer does not modify the input features for the two classification layers, except for adjustments related to the graph itself. In the case of BA-Shapes, all models employ a standard GNN layer to encode all input features.The necessity of this initial pass arises from the fact that the standard GAT implementation alone is incapable of exclusively detecting the graph structure. For instance, in cases where all node feature vectors consist solely of '1', our GAT implementation will aggregate and normalize the surrounding feature vectors, yielding once again a feature vector of '1' for all nodes.

We present the hyperparameters used for controlling the sparsity of all methods employed in this work in Table 3. For SGAT, we set γ to different values depending on the dataset. Specifically, we set $\gamma = 10^{-5}$ for CiteSeer, Cora and BA-Shapes, $\gamma = 10^{-6}$ for PubMed and Amazon Photo, and $\gamma = 10^{-7}$ for Actor.

Table 3: Configuration of hyperparameters used for controlling sparsity.

Method	**Hyperparam.**	**Values**
SGAT	weight of ℓ_0 penalty	$\{0, 1.0, 1.5, 2.2, 2.4, 2.6, 2.8, 3.0, 3.2, 3.4, 3.6, 3.8, 4.0, 5.0, 6.0\} \times\gamma$
DropEdge	portion of dropped edges	{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0}
NeuralSparse	maximum number of edges per node	{1, 2, 4, 6, 8, 10, 12, 16, 20, 25, 50, 100}
Entmax	propensity to sparsity (α)	{1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0, 6.0, 10.0}
Top-k	maximum number of edges per nodes	{1, 2, 4, 6, 8, 10, 12, 16, 20, 25, 50, 100}
MapSelect-L	SparseMAP absolute budget (B)	{1, 2, 4, 6, 8, 10, 12, 16, 20, 25, 50, 100}
MapSelect-G	SparseMAP percentage budget (B)	{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0}

B.2 Metrics

We evaluate interpretability with the sparsity and fidelity metrics proposed by BAGEL (Rathee et al., 2022) and ZORRO (Funke et al., 2023), defined next. These scores are calculated over the explanations of 300 randomly selected nodes. The random selections are kept consistent for each dataset. Both metrics are evaluated against the percentage of *removed edges*, where an edge is considered removed when its explanation score is zero.

Rationale sparsity. Computes the Shannon entropy over the explanation vector $\boldsymbol{p} \in \triangle_{n-1}$:

$$H(\boldsymbol{p}) = -\sum_i p_i \log p_i. \tag{12}$$

RDT-fidelity. Given explanations $\boldsymbol{p_i} \in \triangle_{n-1}$ for each node $1 \leq i \leq n$, let $\boldsymbol{M} \in [0,1]^{n \times n}$ denote a mask matrix, such that $M_{ij} = p_{ij}$. The RDT-Fidelity concerning the network Φ and the noise distribution $\mathcal{N}$, is expressed as follows:

$$\mathcal{F}(\boldsymbol{M}) = \mathbb{E}\left[1_{\Phi(\boldsymbol{X})=\Phi(\tilde{\boldsymbol{X}}(\boldsymbol{M}))}\right], \tag{13}$$

where $\boldsymbol{X} \in \mathbb{R}^{n \times n}$ represents the input, and $\tilde{\boldsymbol{X}}(\boldsymbol{M})$ is a perturbed input defined as:

$$\tilde{\boldsymbol{X}}(\boldsymbol{M}) = \boldsymbol{M} \odot \boldsymbol{X} + (1 - \boldsymbol{M}) \odot \boldsymbol{Z}, \quad \boldsymbol{Z} \sim \mathcal{N}. \tag{14}$$

As Rathee et al. (2022), we set the noise distribution as the global empirical distribution of the input features.

B.3 Datasets

As a node classification task the Cora, PubMed, CiteSeer, Actor and Amazon Photo datasets (Yang et al., 2016; Pei et al., 2020; Shchur et al., 2018) are evaluated using the default configurations provided by PyTorch Geometric (Fey & Lenssen, 2019). These datasets can be classified as *transductive*, indicating that there is no isolation of the training set from the validation set as all data points are part of a single graph. We provide an overview of the datasets in Table 4.

Table 4: Overview of the datasets used in our experiments.

	# nodes	# edges	# features	# classes
Cora	2,708	10,556	1,433	7
CiteSeer	3,327	9,104	3,703	6
PubMed	19,717	88,648	500	3
Actor	7,600	30,019	932	3
Amazon Photo	7,650	238,162	500	3
BA- Shapes	700	3936	1	4

To evaluate our method with ground-truth explanations, we opted for the Barabasi-Albert (BA-Shapes) dataset (Ying et al., 2019). This is a dataset with 300 random nodes and a set of 80 "house"-structured graphs connected to it. The dataset contains 4 classes; a node can be classified as the top, the middle or the bottom of a house or as not being part of a house. Each node has a single input feature equal to 1, *forcing the network to only classify based on the graph structure*. As a ground truth, an edge and node mask are passed containing all edges and nodes that are part of a house-structure.

C Explanation extraction

Node-level explanation. Node-level explanations are generated by (i) setting the value of edge weights as the attention weights of GAT network that produced the original classification, and (ii) propagating an identity matrix of size N, the number of nodes in the graph. We can formally describe this as follows. Let $\tilde{\pi}_{ij}^{(\ell)} \in \mathbb{R}$ be the attention weight associated with the edge between node i and its neighbour j at layer ℓ in the original GAT network, extracted after applying an interpretability method. For example, in MapSelect, $\tilde{\pi}_{ij}^{(\ell)}$ is masked according to SparseMAP's output and then re-normalized, as stated in the left part of Equation 6. Overall, for each explainability method, we perform the following steps to extract node-level explanations:

1. Recover the attention weights $\tilde{\pi}_{ij}^{(\ell)}$ by running the GAT network on the input graph with original feature vectors $\boldsymbol{h}_i \in \mathbb{R}^d$, for each node i.

2. Create a new one-hot vector representation for node i, $\boldsymbol{h}_i^{(0)} = \{0,1\}^N$, where $h_{ij} = 1$ if $i = j$ and $h_{ij} = 0$ otherwise.
3. Propagate the new representation through a weighted-message passing network, with as many layers as the original network. That is, we compute new node features as follows:

$$\boldsymbol{h}_i^{(\ell+1)} = \sum_{j \in \mathcal{N}_i} \tilde{\pi}_{ij}^{(\ell)} \boldsymbol{h}_j^{(\ell)}, \tag{15}$$

   where $\mathcal{N}_i$ represents the set of neighbors of node i. That is, the new node representation is simply a weighted sum of one-hot vectors.
4. Obtain the explanation for node i from its final node features ($\boldsymbol{h}_i^{(\text{final})} \in \mathbb{R}^N$):

$$\boldsymbol{p}_i = \frac{\boldsymbol{h}_i^{(\text{final})}}{\sum_{j=1}^N \boldsymbol{h}_{ij}^{(\text{final})}} \in \triangle_{N-1}, \tag{16}$$

   where p_{ij} represents the importance of node j to the classification of node i. Therefore, to get a final node-level explanation with respect to a target node $i^\star$, we simply extract $\boldsymbol{p}_{i^\star}$.

Note that all nodes outside of the computation graph of node i will receive an importance score of zero. When calculating the fidelity and sparsity scores, these importance scores are not included.

Calculating fidelity and sparsity. The fidelity and sparsity scores are calculated over each extracted node-level explanation. The scores have been computed and averaged for 300 randomly selected nodes. For each dataset, the same 300 nodes were used to evaluate all methods.

D Additional Results

D.1 Real-world datasets

Sparsity-entropy tradeoff. Figure 6 shows the impact of sparsification on entropy (described in §B.2), where a low entropy indicates a more focused rationale. The initial performance of Entmax and top-k can be attributed to a better allocation of the attention distribution. This is seen by the lower entropy of Entmax, indicating a more focused explanation.

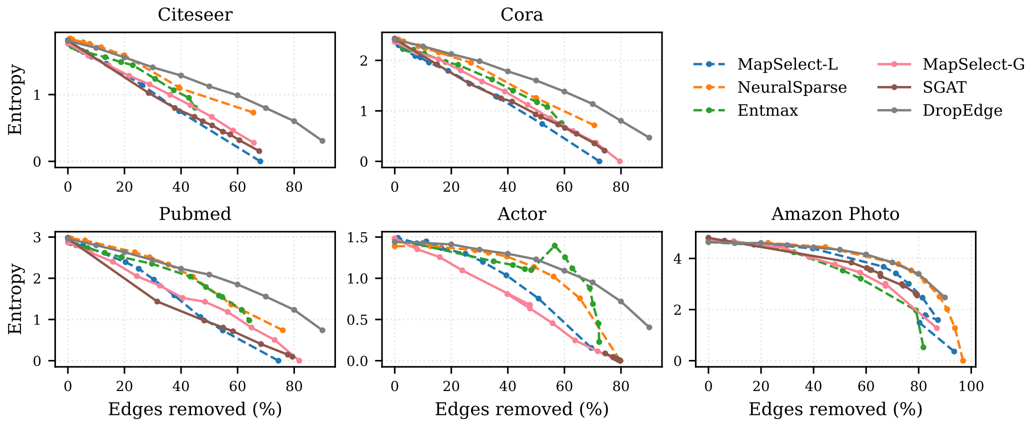


Figure 6: Trade-off between graph sparsity and explanation sparsity.

Sparisty-interpretability tradeoff. Forcing the model to maintain the self-loops (not allowing them to be masked out) greatly improved the fidelity scores as shown in Figure 7. Only Entmax produces a better result when not maintaining the self-loops, however, this approach also presented more instability. As a remark, in the main paper we provide the Entmax version that preserves the self-loops for the sake of consistency.

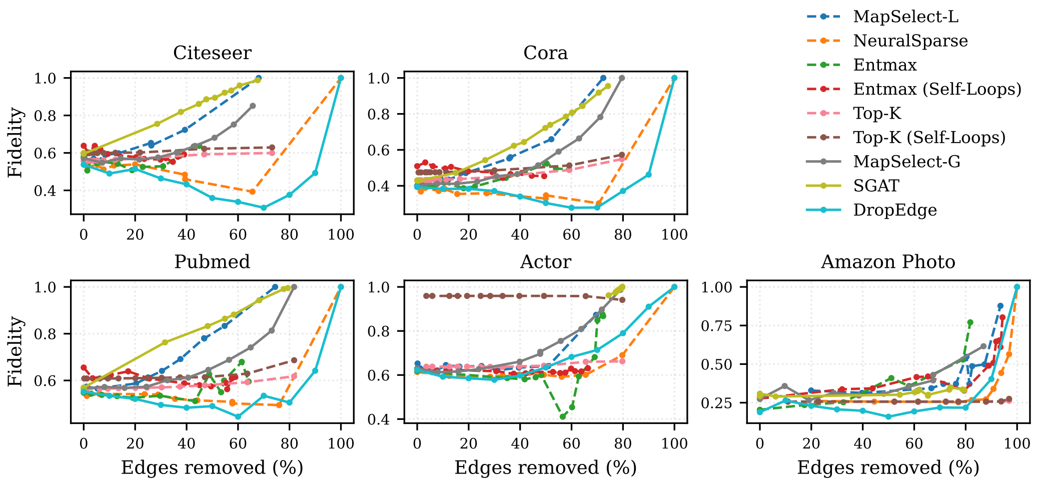
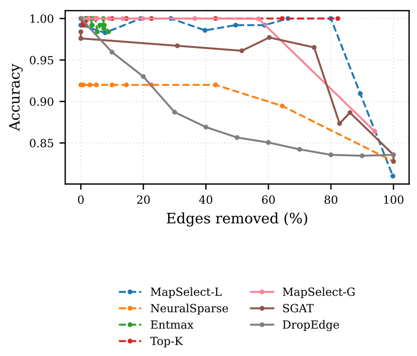


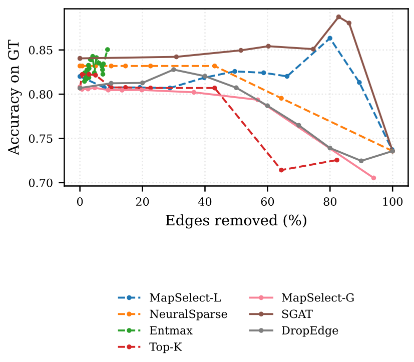
Figure 7: Trade-off between fidelity and sparsity, with and without maintaining self-loops.

D.2 Synthetic dataset

Tradeoffs. In Figure 8a, we illustrate the trade-off between sparsity and performance in the BA-Shapes dataset. Interestingly, even with the removal of all edges, an accuracy of 85% is achieved. This can be attributed to the initial pass through a single GNN layer for all methods, since this layer helps all methods to learn sparse subgraphs. In addition to assessing the AUC score, we also conducted an evaluation of the extracted rationales in relation to the ground truth in terms of raw accuracy, as depicted in Figure 8b. For this, we set a threshold of 0.5 for binarizing explanations. Intuitively, the accuracy metric applies a greater penalty to values approaching zero rather than exactly zero, treating them with the same severity as values that are higher but still fall below the threshold. This accounts for the disparity observed in Figure 4, where the AUC score exhibits differing starting points due to certain models learning weights that approach zero more than other models. Via this accuracy score, the trend of explanations improving as more edges are removed becomes even more pronounced.



(a) Trade-off between sparsity and task accuracy.



(b) Trade-off between sparsity and interpretability (accuracy).

Figure 8: Additional results on the BA-Shapes dataset.

Explanation example. In the BA-Shapes dataset, edges that do not pertain to a “house” structure are considered irrelevant. We anticipate that MapSelect will effectively filter out these non-structural edges, offering the remaining edges as a rationale. As illustrated in Figure 9, a subgraph from the BA-Shapes dataset showcases the attention weights learned in one of our experiments. As anticipated, the majority of attention is directed towards the edges constituting this house-like structure.

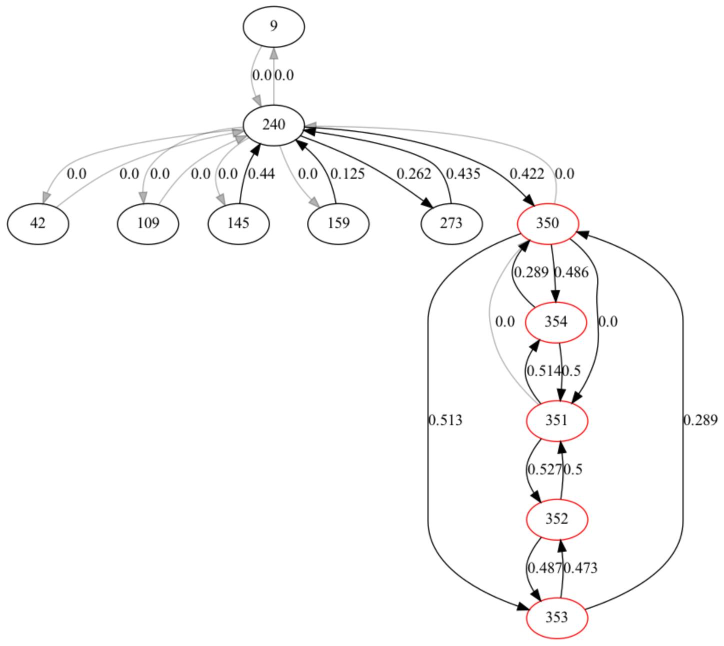


Figure 9: Example of the generated attention values by *MapSelect-L* on the BA-Shapes dataset with a budget of $B = 2$. Here we show all nodes within a k-hop distance of 2 from node 350. A red border indicates that a node is part of a "house".

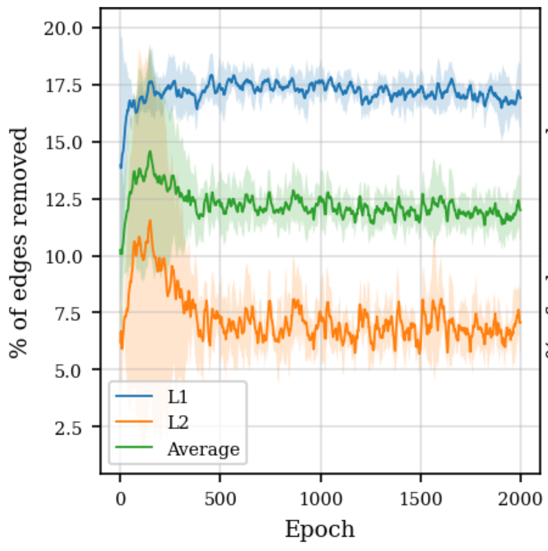
4
Additional Evaluation

As the experiments performed in this research are not limited to those shown in the paper, this section will expand on these results. Additionally, we will discuss a failed concept.

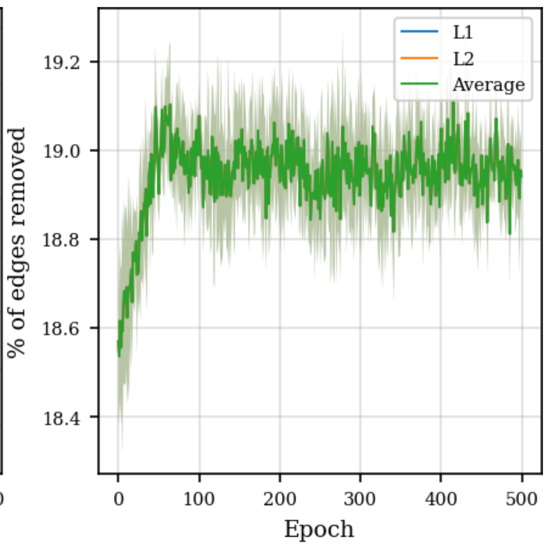
4.1. Additional Experiments

4.1.1. Behaviour of Sparsity during Training

To gain insight into the behaviour of the sparsification over time the percentage of edges removed was plotted against the epochs in Figure 4.1a and Figure 4.1b.
As we can see, in the Entmax setup there is a large difference between the two layers. The first layer maintains a high level of sparsity whereas the second layer removes a significantly lower percentage of edges. Potentially, removing the connection to first-degree neighbours presents a larger gain in performance than removing edges that aggregate the features of second-degree neighbours. Despite the network exhibiting significant variance in its initial 500 epochs, we observe convergence to a stable state as the network continues training beyond this threshold.
The MapSelect setup presents a steady percentage of edges removed from the start, additionally, there is no difference between both layers. This is expected as the rationalization mask is calculated in a separate layer and applied to both layers. As the MapSelect method allows us to force tightly control the number of edges selected, there is little variance in the percentage of edges removed.



(a) Entmax with $\alpha = 2.5$.

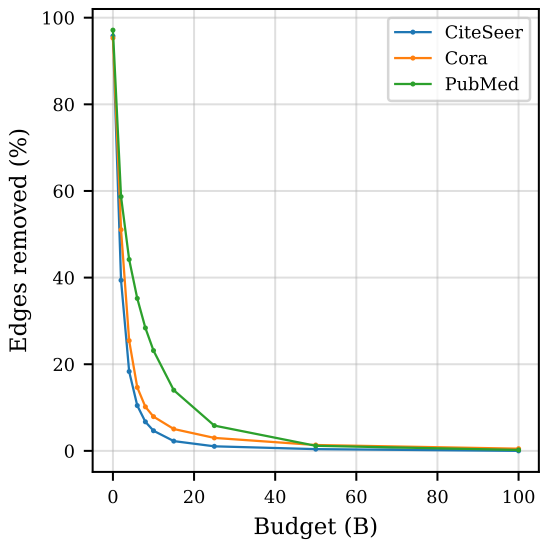


(b) MapSelect-L with a relative budget of 0.7.

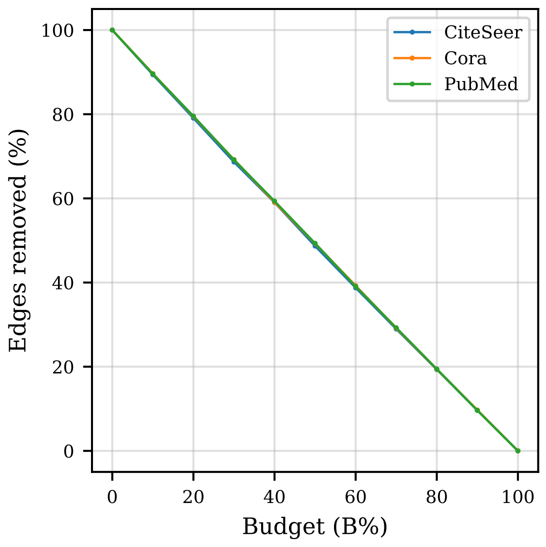
Figure 4.1: The development of sparsity during training. In the MapSelect case, both layers present exactly the same sparsity.

4.1.2. Control over the Sparsity

The main advantage of the MapSelect method is the flexibility and control over the sparsity in the network. As presented in Figure 4.2, both MapSelect methods present strong control over the amount of edges removed. The data points follow a consistent pattern and have low variance across evaluations. Note that the budget for MapSelect-L is set for each neighbourhood, so there is no linear relationship.



(a) MapSelect-L



(b) MapSelect-G

Figure 4.2: The number of edges removed for the corresponding sparsity setting.

.

The Entmax method, presented in Figure 4.3, is harder to control. There is a notable variance across evaluations and the data points do not follow a consistent path. The α parameter does not have a straightforward connection to the number of edges to be removed and necessitates computation to assess this relationship.

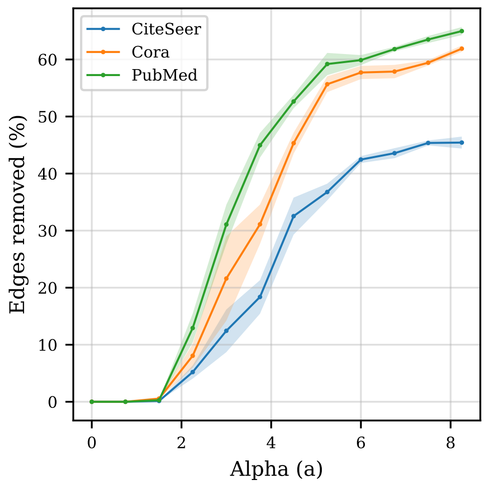


Figure 4.3: The number of edges removed for the corresponding setting for α in α-entmax.

The baseline SGAT was even harder to control. Finding the correct parameters for the SGAT method presented a challenge as there was a lot of variation during training time and accross the different evaluations at test time. Each dataset required separate calculations of parameters due to large differences in the domain for λ. CiteSeer and Cora differed from PubMed by an order of magnitude (Figure 4.4).

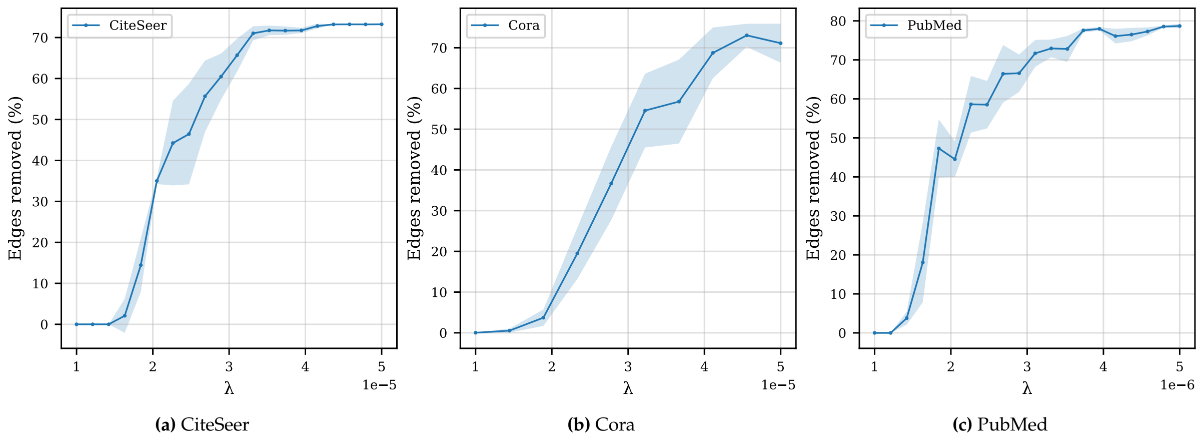


Figure 4.4: The number of edges removed for the corresponding setting for λ in SGAT.

4.1.3. Ad-hoc Control over the Sparsity

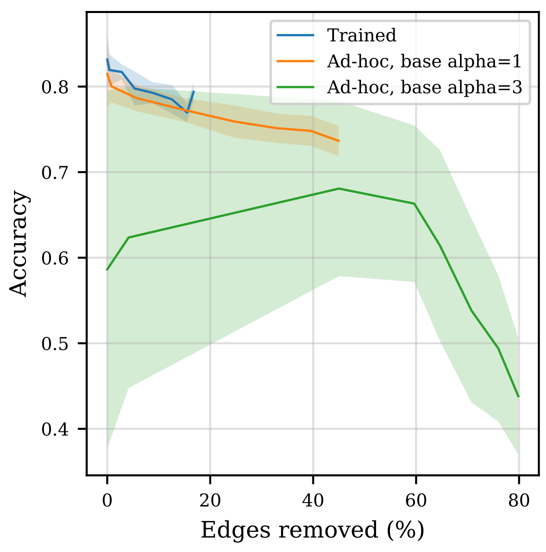
A potential advantage of our method would be the ability to change the sparsity of the subgraph after training the network. This would allow for determining the sparsity of your classification after training, something that is not allowed by methods that prune the graph through regularization like SGAT. To evaluate this, we compare the performance of the model when trained with a specific parameter ϕ against a model that was trained on a different parameter $\hat{\phi}$ and changed to ϕ after training. In Figure 4.5 these results are presented, the evaluation was performed on Cora only as the idea is not further explored in this report. As 'base' models, the model that was trained with parameter $\hat{\phi}$, we trained one with a very low sparsity and one with a very high sparsity setting.
Regarding the Entmax case, stable results are presented when setting the sparsity parameter after training using the base model with $\hat{\alpha} = 1$. Additionally, the performance is on par with the trained model, though it is slightly outperformed. The model with a base of $\hat{\alpha}$ is very unstable and presents a large decrease in performance. In both ad-hoc cases, a higher percentage of edges removed is achieved using the same α setting. This is explained by the network not converging the attention scores to a smaller range so that no scores are set to absolute zero by the α-entmax function.
In the MapSelect case, all settings present similar results. This is expected as the SparseMAP function will always enforce the preset budget. Interestingly, there is no large difference in performance between all three setups. However, the model trained with the low budget of $\hat{B} = 4$ does present more instability. Surprisingly the model trained on a high budget of $\hat{B} = 100$ performs better for a sparsity beyond 40% of edges removed.

The effect on interpretability was not studied due to time constraints and left for further work. Intuitively, we would anticipate that the fidelity score of an ad-hoc model is lower compared to the trained model. This is because the trained model is trained to detect irrelevant edges, yielding better explanations. However, we would also expect more instability due to the SparseMAP and α-entmax functions not having trained with sparse distributions in the ad-hoc case, something that is not represented by the ad-hoc models trained with a low sparsity setting as a base.

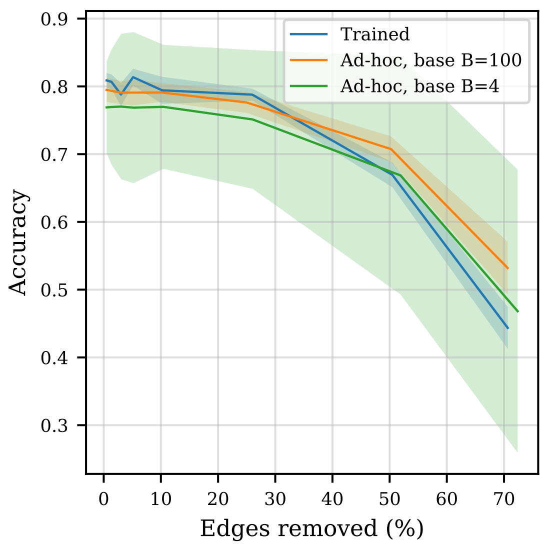
4.2. Failed Concepts

4.2.1. MapSelect-M

As presented by [10], the SparseMAP function allows for optimization over a matrix input. This allows for matching different elements to each other based on their shared score (Figure 4.6). Intuitively, this works for two “sentences” as presented in Figure 4.7. Extending this idea to a GNN can allow for a novel way to extract a fitting rationale for a single node. Additionally, the constraints provided by the SparseMAP would allow for a wide ability to manipulate the local attention structure. However, implementing this idea is not trivial as, in contrast to the α-entmax function, it does not present a suitable replacement for the softmax function. The softmax function expects a 1D tensor whereas the SparseMAP function can solve for a 2D matrix. This matching approach would switch the edge



(a) Pre-trained Entmax model against ad-hoc models trained with $\alpha = 1$ and $\alpha = 3$.



(b) Pre-trained MapSelect-L model against ad-hoc models trained with $B = 4$ and $B = 100$.

Figure 4.5: Comparison between models with preset sparsity and ad-hoc sparsity on the Cora dataset.

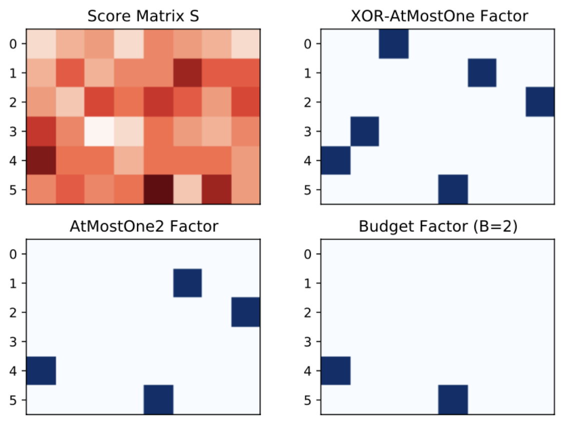


Figure 4.6: A plot by [10] of the selected matchings by the SparseMAP function based on the score matrix S and different constraints.

selection from the node perspective, where all edges connected to the target node are considered, to the edge perspective, where all scores in the full graph or a local structure a considered. The nodes in this structure are set as the x and y components and the attention coefficients assigned to their connecting edges are represented as scores in the matrix. SparseMAP will solve for the given matrix and only select those edges that yield the highest result according to the constraints. Two approaches are considered.

Global Matching The most simple approach is to construct an NxN adjacency matrix containing the attention score for each edge present in the graph and feed this matrix directly to the SparseMAP function. More formally, let $\mathbf{\Pi} \in \mathbb{R}^{N \times N}$ be a matrix that represents the stack of the attention scores $\boldsymbol{\pi}_i^{(1)}$ of the 1st GAT layer for all $1 \leq i \leq N$ nodes and $1 \leq j \leq N$ nodes, padding (i, j) positions with 0 when $(i, j) \notin \mathcal{G}$. Defined as:

$$\mathbf{\Pi} = \begin{bmatrix} \pi_{11} & \pi_{12} & \dots & \pi_{1N} \\ \pi_{21} & \pi_{22} & \dots & \pi_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \pi_{N1} & \pi_{N2} & \dots & \pi_{NN} \end{bmatrix}, \quad \boldsymbol{P} = \text{SparseMAP}(\mathbf{\Pi}/t; B). \tag{4.1}$$

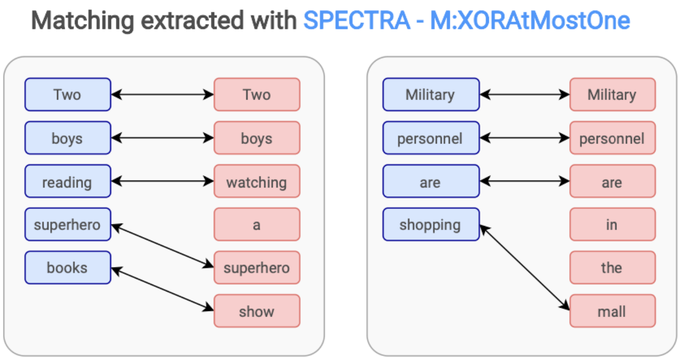


Figure 4.7: An example by [10] of how the SparseMAP function can be used to match words in two sentences. An XOR constraint and AtMostOne constraint are applied.

The zero values in the mask matrix $\boldsymbol{P} \in [0,1]^{N \times N}$ will be used to deactivate edges, and as a result, they will not contribute towards the final decision.
However, when testing this method it did not complete within a reasonable time (< 8 hours). Hence, this approach was abandoned.

Local Matching This computation problem is potentially solved by running the SparseMAP function only over the computation graph of the target node. If node i is the target node, all nodes within a k-hop neighbourhood, k being equal to the number of GAT layers, will form the computation graph. This subgraph represents the only nodes that impact the classification of node i. This approach assumes that the full graph is not dense, this would result in subgraphs that approach the size of the full graph and will therefore not compute within a reasonable time.
Implementing this approach requires splitting the original graph into N computations graphs such that a separate sparse matching can be performed for each node, formally denoted as:

$$\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i) \tag{4.2}$$

where $\mathcal{V}_i$ represents the set of nodes within the k-hop neighbourhood of node i in the original graph $\mathcal{G}$ and $\mathcal{E}_i$ is the set of directed edges within the k-hop neighbourhood of node i. The rest of the calculation is the same as presented in Equation 4.1, but performed separately over all subgraphs $\mathcal{G}_i \in \mathcal{G}$ where $i \in \{1, \ldots, N\}$.
Sadly, this method did not perform well enough to further explore this idea. The lack of performance can be explained as follows, when a computation graph $\mathcal{G}_i$ of node i is passed through the network, the attention scores attributed to the edges in this subgraph will not differ from when the full graph is passed through the network, as the same features for the attention mechanism are used as input. This can result in a situation where only the edges will be selected that connect nodes in the subgraph that are not the target node. This leaves no edges connected to the target node, resulting in poor classification performance. Additionally, as attention scores are normalized per target node, the edges connected to a node with a low degree will be favoured over nodes with a high degree.
Whereas this last issue can be solved by using unnormalized scores, the first issue is harder to solve and is left for further work. Potentially, adding the target node i to the attention score calculation for edge $(j,k) \in \mathcal{E}_i$, resulting in $e_{jk} = g_\phi(n_j, n_k, n_i)$, would solve this issue by always partially attending to the target node. Here, $g_\phi : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ represents the learnable attention score function. This idea is left for future work.

5 Additional Related Work

In this section, we discuss two topics that received little attention within the paper, primarily because of space limitations. Initially, we touch upon the question of whether attention actually serves as a viable model explainer. Following that, we conduct a more in-depth exploration of alternative methods for achieving self-interpretability in Graph Neural Networks (GNNs).

5.1. Is Attention Interpretable?

Given that a comprehensive evaluation of attention interpretability in Graph Attention Networks (GATs) is lacking, we must turn to insights from related fields to assess the interpretability of GATs. The concept of attention originally emerged in Natural Language Processing (NLP) as a mechanism for models to autonomously determine which features warrant focus to optimize task performance [8]. This innovation not only enhanced model performance but also enabled researchers to derive inherent attention-based explanations by extracting these attention weights. While these attention weights can shed light on features deemed relevant by the model, they attend to hidden representations of the original input, sparking a pertinent debate.

In [31] and [37] experiments are performed to determine whether attention weights can be considered as accurate model predictors. They argue that (i) attention weights should correlate with feature importance similar to the gradient-based method and (ii) alternative attention weights (counterfactual) should lead to changes in the prediction. After performing the experiments, [31] concluded that attention can by no means function as a fail-safe indicator. These results are backed up by *Attention is not explanation* [13], where they show that learned attention weights frequently do not correlate with gradient-based measures of feature importance and entirely different attention distribution can be found yielding the same result. [37] disagrees, and shows that attention weights are interpretable and correlate to feature importance in cases where attention weights are essential for the model's prediction. Furthermore, *Attention is not not explanation* [41] counters the arguments made in [13], by stating that the definition of an explanation is crucial in answering the question. They indicate that attention-based explanations can be considered plausible. However, as multiple possible attention distributions can yield the same result, the explanation can not always be considered faithful.

Specifically relevant to our research, [19] discusses whether inducing sparse attention increases model interpretability. Against expectations, they observe a decrease in the correlation between the attention distribution and input feature importance measures, concluding that sparse attention does not enhance model interpretability. Whereas, in *The Explanation Game* [35] selective attention (α-entmax) is implemented to produce sparse attention, providing superior usefulness over gradient-based methods for the proposed tool to provide explanations for model decisions.

In summary, the debate surrounding whether attention serves as a reliable model predictor persists. However, it can be confidently asserted that this reliability is contingent upon context—factoring in the complexity of the model, the specific use case, and one's definition of explainability.

5.2. Other Approaches to Interpretable GNNs

When it comes to self-interpretable models [14], sparse attention is just one of the available options. There are other methods that apply a structural constraint to the computation graph to enhance the model's interpretability

One of these methods is DIR [42], a method that aims to discover an invariant rationale by conducting interventions on the training distribution to create multiple interventional distributions. Then, the causal rationales that are invariant across different distributions are approached by filtering out the spurious patterns that are unstable.

Another method, called ProtoGNN [4], incorporates prototype learning within GNNs. Prototype learning, a type of case-based reasoning, facilitates predictions for novel instances by measuring their similarity to a set of previously learned exemplar cases known as prototypes to aid the prediction. In the context of ProtoGNN, it calculates similarity scores between the graph embedding and multiple learned subgraph prototypes. Prototypes exhibiting significant similarity can then be employed as explanations.

SEGNN [20] is a framework that can find k-nearest labelled nodes for each unlabeled node to give explainable node classifications, where the nearest labelled nodes are found by an interpretable similarity module in terms of both node similarity and local structure similarity. These k-nearest nodes can be used to derive an explanation subgraph.

Another approach is called KER-GNN [6]. Inspired by convolutional filters in Convolutional Neural Networks (CNNs), this work adopts trainable hidden graphs as graph filters. These are combined with subgraphs to update node embeddings using graph kernels. The output of these filters can be used to highlight important substructures in the graph.

As a final work, we discuss L2XGNN [30], a framework explicitly engineered to offer accurate explanations. L2XGNN constructs a mechanism for identifying explanatory subgraphs, which are exclusively integrated into the message-passing operations of Graph Neural Networks (GNNs). L2XGNN possesses the capability to enforce particular characteristics within the chosen subgraph, such as ensuring it is both sparse and connected.

Only SEGNN and KER-GNN have been evaluated on node classification datasets. All other methods have been developed for graph classification tasks.

6 Conclusion

6.1. Thesis Summary

Graph Attention Networks (GATs) have shown remarkable performance in capturing complex graph structures by assigning dense attention weights over all neighbours of a node. Attention can act as an inherent explanation for the model output, highlighting those neighbours that are most important. However, in practice, the dense nature of the attention layer causes a lack of focus, undermining model interpretability. In this work, we introduced MapSelect and Entmax, two novel methods offering controlled sparse attention. For MapSelect, we proposed two distinct approaches: a local approach maintaining a fixed subgraph size per node, and a global approach preserving a percentage of the full graph. In Chapter 3, we performed a comprehensive evaluation of several baselines to our proposed methods on five real-world datasets. For each model, its performance and interpretability are evaluated for increasingly sparse configurations, yielding Pareto curves that provide insights into the performance-interpretability trade-off for all datasets. We show that MapSelect is able to produce highly competitive results and stands out on the most dense dataset. Additionally, we conduct a validation process on all models using a synthetic dataset. This allowed us to validate the performance of our generated explanations in comparison to the established ground truth explanations for the dataset. In Chapter 4, we extended our research by investigating the behaviour of sparsity within the proposed methods and the effect of changing the sparsity parameter after training. We concluded that altering sparsity after training in a non-sparsified environment still produces competitive outcomes, whereas altering sparsity after training with significant sparsification leads to unpredictable results.

6.2. Answer to Research Questions

In this section, we will discuss the research question posed in Chapter 1:

RQ. **How can we produce interpretable Graph Attention Networks through sparse attention?**

As Graph Attention Networks leverage the attention mechanism to highlight the most relevant edges in the input graph, these attention weights can be used as an inherent explanation of the model prediction. Through the application of sparse attention, we can generate an even more focused attention mask by entirely removing irrelevant edges. Removing edges from the input graph will create a sparse subgraph, facilitating easier interpretation.

We examine two existing works, NeuralSparse [15], and SGAT [43], which employ sparse attention. We classify NeuralSparse as a local sparsification method, as it selectively removes edges within each neighbourhood using top-k sampling via the Gumbel-Softmax. In contrast, SGAT is labelled as a global approach because it specifies edges over the full graph through the use of a regularization term. We also introduce a local baseline method called top-k, which involves selecting only the top-k edges based on their respective scores. As for the global baseline, we evaluate DropEdge, a technique in which edges are randomly chosen and removed from the entire graph. It's worth noting that both DropEdge and top-k do not incorporate training with a sparse attention mechanism, setting them apart from the aforementioned approaches.

We propose two sparse attention methods. The first method employs the sparse inference technique known as α-entmax, serving as a replacement for the softmax. α-entmax operates similarly to softmax but precisely sets scores that are close to zero to zero. By employing this method, we can implement sparse attention in the conventional GAT by simply substituting the softmax. While this parameter allows for user-controlled sparsification, it is not possible to determine in advance how many edges will be retained during the classification task. The second method leverages the sparse inference method SparseMAP and produces both a local and global approach, called MapSelect-L and MapSelect-G respectively. In this approach, we add a separate GAT layer that is only used to generate a binary mask by feeding the attention scores to the SparseMAP function and applying this binary mask over the input graph used in the subsequent layers. In both methods, we gain significant performance in both accuracy and interpretability when maintaining the self-loops.

In Chapter 3, we investigate the influence of an increased sparsification on both model performance and interpretability. Through this analysis, we can assess the trade-off between performance and interpretability. Entmax yields suboptimal results and is included in the evaluation as a baseline. MapSelect-L presents state-of-the-art results in terms of accuracy and interpretability with regard to its respective baselines. MapSelect-G stands as the singular method capable of providing global control over the sparse subgraph while concurrently achieving competitive results in both accuracy and interpretability. Both MapSelect methods present themselves as the only methods to consistently improve interpretability on all datasets.

In Chapter 4, we conducted supplementary experiments. Our investigation into sparsity behaviour revealed a notable distinction between the MapSelect and Entmax methods. Specifically, the Entmax method exhibited a gradual convergence in the number of removed edges over an extended duration, whereas MapSelect maintained a nearly constant percentage of edges removed throughout the entire training process. Notably, Entmax exhibited a higher degree of edge removal in the second layer compared to the first, suggesting that second-degree neighbours may hold less significance. Moreover, our examination of ad-hoc sparsity adjustments showed only a marginal decline in performance when modifying the sparsity configuration in a network initially trained without any form of sparsification.

To conclude, we identified several important model features for achieving the most optimal performance-interpretability trade-off in a Sparse & Interpretable Graph Attention Network:

1. **Maintaining Self-Loops**: We found that preserving self-loops during classification had a consistently positive impact on all models.
2. **Global vs. Local Sparsification**: Global sparsification outperformed local sparsification in terms of accuracy and interpretability, even when a similar number of edges were removed. Only on the most dense dataset, a local approach outperforms global approaches.
3. **Ad-hoc Sparsification**: It is strongly recommended to train the model with the desired sparsification level to achieve both steady and high performance. However, it is possible to maintain competitive performance when inducing sparsity in a network that has not been trained on a sparse subgraph.
4. **Convergence of Attention Scores**: Allowing attention scores to slowly converge to zero before removal, as opposed to fixed node removal, led to a more consistent convergence and higher interpretability in most cases. Nevertheless, with the current implementations, this will limit the control over the number of edges removed.

Additionally, we identified a potential beneficial characteristic:

- **Separating Sparsification from Attention Mechanism**: Having the sparsification layer or function distinct from the attention mechanism, as implemented by SGAT and MapSelect, showed promise. However, further evaluation is needed to conclusively establish this as a beneficial feature. Additionally, having a single or multiple subsequent attention layers to apply the generated mask to did not impact the results.

When implementing such a network, users will encounter design choices related to control over the subgraph. It is worth noting that in some cases, a locally controlled method may provide slightly lower performance but deliver explanations with a predictable node count. This predictability can enhance the value of the generated explanations.

6.3. Future Work

This is the first research on the SparseMap function in the context of GNNs. There are many interesting research paths to explore from here. In this section, we will touch on a few that directly tie into our work.

Towards SGAT SGAT has presented itself as the most optimal implementation in most cases. We attributed this performance to the more loosely controlled approach taken by this method. Potentially, this can also be created in the SparseMAP model. A slowly decreasing temperature or budget might allow the model to remove edges in a more stable way, yielding a positive effect with regard to interpretability.

Increase structural constraints L2XGNN [30] is a method that imposes strong constraints on a subgraph that will act as a rationale. This was the initial aim of the failed matching method proposed in Section 4.2.1. The explanation of why the matching method fails on node classification tasks is not valid for graph classification tasks. It would be interesting to investigate the performance of the MapSelect matching method on a graph classification case. If the approach proves successful in a graph classification task, SparseMAP's capacity to meet specific structural constraints in its output can be effectively utilized.
Furthermore, additional structural constraints carry the potential advantage of overcoming over-smoothing [27] and over-squashing [1].

Speed up An increase in training time prevented us from evaluating the largest datasets. The principal contributing factor to the observed decrease in speed was identified to be the custom scatter operation, rather than SparseMAP itself. This means the training time for MapSelect-L can be greatly reduced by implementing a tailored scatter method that allows quick budget assignment over the full tensor in C++.

References

[1] Uri Alon and Eran Yahav. “On the Bottleneck of Graph Neural Networks and its Practical Implications”. In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=i80OPhOCVH2.

[2] Edward A Codling, Michael J Plank, and Simon Benhamou. “Random walk models in biology”. In: *Journal of the Royal society interface* 5.25 (2008), pp. 813–834.

[3] John M Danskin. “The theory of max-min, with applications”. In: *SIAM Journal on Applied Mathematics* 14.4 (1966), pp. 641–664.

[4] Yanfei Dong et al. *ProtoGNN: Prototype-Assisted Message Passing Framework for Non-Homophilous Graphs*. 2022.

[5] Finale Doshi-Velez and Been Kim. *Towards A Rigorous Science of Interpretable Machine Learning*. 2017. arXiv: 1702.08608 [stat.ML].

[6] Aosong Feng et al. “KerGNNs: Interpretable Graph Neural Networks with Graph Kernels”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36.6 (June 2022), pp. 6614–6622. DOI: 10.1609/aaai.v36i6.20615. URL: https://ojs.aaai.org/index.php/AAAI/article/view/20615.

[7] T. Funke et al. “Zorro: Valid, Sparse, and Stable Explanations in Graph Neural Networks”. In: *IEEE Transactions on Knowledge & Data Engineering* 35.08 (Aug. 2023), pp. 8687–8698. ISSN: 1558-2191. DOI: 10.1109/TKDE.2022.3201170. URL: https://doi.ieeecomputersociety.org/10.1109/TKDE.2022.3201170.

[8] Andrea Galassi, Marco Lippi, and Paolo Torroni. “Attention in natural language processing”. In: *IEEE transactions on neural networks and learning systems* 32.10 (2020), pp. 4291–4308.

[9] Hongyang Gao and Shuiwang Ji. “Graph U-Nets”. In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 2083–2092. URL: https://proceedings.mlr.press/v97/gao19a.html.

[10] Nuno M. Guerreiro and André F. T. Martins. “SPECTRA: Sparse Structured Text Rationalization”. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 6534–6550. DOI: 10.18653/v1/2021.emnlp-main.525. URL: https://aclanthology.org/2021.emnlp-main.525.

[11] Will Hamilton, Zhitao Ying, and Jure Leskovec. “Inductive representation learning on large graphs”. In: vol. 30. 2017.

[12] Alon Jacovi and Yoav Goldberg. “Aligning Faithful Interpretations with their Social Attribution”. In: *Transactions of the Association for Computational Linguistics* 9 (Mar. 2021), pp. 294–310. ISSN: 2307-387X. DOI: 10.1162/tacl_a_00367. eprint: https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00367/1923972/tacl_a_00367.pdf. URL: https://doi.org/10.1162/tacl%5C_a%5C_00367.

[13] Sarthak Jain and Byron C. Wallace. “Attention is not Explanation”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 3543–3556. DOI: 10.18653/v1/N19-1357. URL: https://aclanthology.org/N19-1357.

[14] Jaykumar Kakkad et al. “A Survey on Explainability of Graph Neural Networks”. In: *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering* (2023). URL: http://sites.computer.org/debull/A23june/p35.pdf.

[15] Thomas Kipf et al. "Neural Relational Inference for Interacting Systems". In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 2688–2697. URL: `https://proceedings.mlr.press/v80/kipf18a.html`.

[16] Transport for London | Every Journey Matters. *Tube — tfl.gov.uk*. `https://tfl.gov.uk/modes/tube/`. [Accessed 24-08-2023].

[17] Dongsheng Luo et al. "Learning to Drop: Robust Graph Neural Network via Topological Denoising". In: *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. WSDM '21. Virtual Event, Israel: Association for Computing Machinery, 2021, pp. 779–787. ISBN: 9781450382977. DOI: `10.1145/3437963.3441734`. URL: `https://doi.org/10.1145/3437963.3441734`.

[18] Andre Martins and Ramon Astudillo. "From Softmax to Sparsemax: A Sparse Model of Attention and Multi-Label Classification". In: *Proceedings of The 33rd International Conference on Machine Learning*. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1614–1623. URL: `https://proceedings.mlr.press/v48/martins16.html`.

[19] Clara Meister et al. "Is sparse attention more interpretable?" In: *arXiv preprint arXiv:2106.01087* (2021).

[20] Lin Meng, Haoran Yang, and Jiawei Zhang. "Stage Evolving Graph Neural Network based Dynamic Recommendation with Life Cycles". In: *2022 International Joint Conference on Neural Networks (IJCNN)*. 2022, pp. 01–08. DOI: `10.1109/IJCNN55064.2022.9892418`.

[21] Tim Miller. "Explanation in artificial intelligence: Insights from the social sciences". In: *Artificial Intelligence* 267 (2019), pp. 1–38. ISSN: 0004-3702. DOI: `https://doi.org/10.1016/j.artint.2018.07.007`. URL: `https://www.sciencedirect.com/science/article/pii/S0004370218305988`.

[22] Vlad Niculae and Andre Martins. "LP-SparseMAP: Differentiable Relaxed Optimization for Sparse Structured Prediction". In: *Proceedings of the 37th International Conference on Machine Learning*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 13–18 Jul 2020, pp. 7348–7359. URL: `https://proceedings.mlr.press/v119/niculae20a.html`.

[23] Vlad Niculae et al. "SparseMAP: Differentiable Sparse Structured Inference". In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 3799–3808. URL: `https://proceedings.mlr.press/v80/niculae18a.html`.

[24] Antonio Ortega et al. "Graph signal processing: Overview, challenges, and applications". In: *Proceedings of the IEEE* 106.5 (2018), pp. 808–828.

[25] Ben Peters, Vlad Niculae, and André F. T. Martins. "Sparse Sequence-to-Sequence Models". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 1504–1519. DOI: `10.18653/v1/P19-1146`. URL: `https://aclanthology.org/P19-1146`.

[26] Mandeep Rathee et al. "BAGEL: A Benchmark for Assessing Graph Neural Network Explanations". In: *arXiv preprint arXiv:2206.13983* (2022). URL: `https://arxiv.org/abs/2206.13983`.

[27] Mandeep Rathee et al. "Learnt sparsification for interpretable graph neural networks". In: *arXiv preprint arXiv:2106.12920* (2021). URL: `https://arxiv.org/abs/2106.12920`.

[28] Yu Rong et al. "DropEdge: Towards Deep Graph Convolutional Networks on Node Classification". In: *International Conference on Learning Representations*. 2020. URL: `https://openreview.net/forum?id=Hkx1qkrKPr`.

[29] Franco Scarselli et al. "The graph neural network model". In: *IEEE transactions on neural networks* 20.1 (2008), pp. 61–80.

[30] Giuseppe Serra and Mathias Niepert. "Learning to Explain Graph Neural Networks". In: *arXiv preprint arXiv:2209.14402* (2022).

[31] Sofia Serrano and Noah A. Smith. "Is Attention Interpretable?" In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 2931–2951. DOI: `10.18653/v1/P19-1282`. URL: `https://aclanthology.org/P19-1282`.

[32] Hamed Shirzad et al. "Exphormer: Sparse Transformers for Graphs". In: *Proceedings of the 40th International Conference on Machine Learning*. Vol. 202. Proceedings of Machine Learning Research. PMLR, 23–29 Jul 2023, pp. 31613–31632. URL: https://proceedings.mlr.press/v202/shirzad23a.html.

[33] Rakshith S Srinivasa et al. "Fast graph attention networks using effective resistance based graph sparsification". In: *arXiv preprint arXiv:2006.08796* (2020). URL: https://arxiv.org/abs/2006.08796.

[34] Naftali Tishby and Noga Zaslavsky. "Deep learning and the information bottleneck principle". In: *2015 ieee information theory workshop (itw)*. IEEE. 2015, pp. 1–5.

[35] Marcos Treviso and André F. T. Martins. "The Explanation Game: Towards Prediction Explainability through Sparse Communication". In: *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*. Online: Association for Computational Linguistics, Nov. 2020, pp. 107–118. DOI: 10.18653/v1/2020.blackboxnlp-1.10. URL: https://aclanthology.org/2020.blackboxnlp-1.10.

[36] Constantino Tsallis. "Possible generalization of Boltzmann-Gibbs statistics". In: *Journal of Statistical Physics* (1988).

[37] Shikhar Vashishth et al. *Attention Interpretability Across {NLP} Tasks*. 2020. URL: https://openreview.net/forum?id=BJe-_CNKPH.

[38] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[39] Petar Veličković et al. "Graph Attention Networks". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=rJXMpikCZ.

[40] Martin J Wainwright, Michael I Jordan, et al. "Graphical models, exponential families, and variational inference". In: *Foundations and Trends® in Machine Learning* 1.1–2 (2008), pp. 1–305.

[41] Sarah Wiegreffe and Yuval Pinter. "Attention is not not Explanation". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 11–20. DOI: 10.18653/v1/D19-1002. URL: https://aclanthology.org/D19-1002.

[42] Yingxin Wu et al. "Discovering Invariant Rationales for Graph Neural Networks". In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=hGXij5rfiHw.

[43] Yang Ye and Shihao Ji. "Sparse graph attention networks". In: *IEEE Transactions on Knowledge and Data Engineering* (2021). URL: https://ieeexplore.ieee.org/document/9399811.

[44] Zhitao Ying et al. "GNNExplainer: Generating Explanations for Graph Neural Networks". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/d80b7040b773199015de6d3b4293c8ff-Paper.pdf.