Mitigating Practical Limitations of GNNs through Explicit Inductive Biases

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Pablo Sánchez Martín, M. Sc.
aus Madrid, Spanien

Tübingen
2024

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation: 21.01.2025

Dekan:	Prof. Dr. Thilo Stehle
1. Berichterstatterin:	Prof. Dr. Isabel Valera
2. Berichterstatter:	Prof. Dr. Robert C. Williamson

Disclaimer: this thesis uses Felix Dangel's doctoral thesis template, which is based on Federico Marotta's `kaobook` template.

Acknowledgments

First and foremost, I am very thankful to my advisor, Isabel Valera. Your continuous support has been fundamental in the last years. Your patience, guidance, and invaluable career and life advice have been key factors throughout this journey. Thanks to you, I have not only improved my scientific skills but also developed essential soft-skills and leadership abilities that I could not have imagined before. I am sure they are going to open many doors for me in the future. Thank you so much.

I am also very grateful to Ulrike von Luxburg, Caterina De Bacco, and Sonja Utz for their role in my thesis advisory committee. Sonja, I really appreciate everything I have learned from you on social media research, especially on ambient awareness and I genuinely appreciate your support in all my decisions. Caterina, thank you also for your remote support on administrative and management tasks from the MPI.

I also want to express my gratitude to the Cluster of Excellence for funding my PhD and the IMPRS-IS administrative staff for all their support along the way and the organization of excellent scientific events I had the pleasure to attend.

I am incredibly grateful to all my collaborators for the amount of ideas exchanged and the invaluable lessons learned. Working alongside more theoretical and practical profiles has truly broadened my research skills.

A huge shoutout goes to all the members of the Probabilistic Machine Learning (PML) group. The time spent with this amazing group has been wonderful. It has been a place where I have felt fully integrated, constantly supported, and where not only cutting-edge research but also fantastic memories were made. Within the PML group, I want to give a special mention to Adrian Javaloy and Batuhan Koyuncu. Not only have we collaborated closely, but we have also built good friendships that extend beyond the office walls. You both have brought support and joy in the last few years.

From the UC3M, I want to extend a big thank you to Pablo M. Olmos, my bachelor's thesis supervisor, who opened the door to the world of research for me and provided full support in my search for opportunities abroad. Pablo, your guidance has been invaluable and has set me on this incredible journey. Also huge thanks to Fernando Perez-Cruz for all his guidance during my master's thesis and as I ventured into the PhD phase. I also want to express my gratitude to Ignacio Peis and Pablo Moreno. They helped me navigate through the research world on my very first days, sharing unforgettable moments and trips along the way. And a special thanks to Pablo Moreno for easing my arrival to Tuebingen, already introducing me to wonderful people there.

I also want to extend my thanks to all the friends I have made in both Saarbrücken and Tübingen. The friendships I have created there are truly special, and I am confident they will last a lifetime. They have brought immense happiness, support, and a sense of home during moments of homesickness. A warm thank you also goes out to my pre-PhD friends. Despite the distance, our trips and annual gatherings have been a constant source of encouragement and motivation. Your support has been very important to me.

Finally, but certainly not least, I want to extend my deepest gratitude to my family: my grandparents, aunts, uncles, cousins, and especially my parents, Maria Jesus and Jose Luis. From the moment I can remember, they have supported me in all my "crazy" adventures, and this time has been

no different. Being far from them has not been easy, but their constant check-ins and full support have meant everything to me. They have been there for me through the highs and lows, offering encouragement every step of the way. Thank you for everything. I dedicate this achievement to you.

Pablo Sánchez Martín

Saarbrücken, March 4, 2024

Abstract

Graphs are a powerful data representation to model complex interactions between entities. They are ubiquitous across various domains, such as social networks, molecular biology, and traffic modeling. Message-passing Graph Neural Networks (GNNs) have recently emerged as effective tools for graph learning, demonstrating state-of-the-art performance in applications including drug discovery, weather forecasting, and recommender systems. Nevertheless, they suffer from both theoretical and practical limitations. Theoretical analyses have revealed that the expressive power of GNN architectures is constrained by the WL-1 isomorphism test [99, 156], indicating an inability to distinguish certain graph structures. Their expressivity is also limited by issues such oversmoothing [121] and oversquazing [5, 31], which in practice, impose limitations on architecture complexity, for instance, the number of layers. Additionally, theoretical and empirical results—including those in this thesis and previous works—expose that the performance of GNNs depends on the task at hand and nature of the graph. Practical challenges further limit the efficacy of GNNs, including limited data availability, local optima in stochastic gradient descent, and a vast hyperparameter space. Consequently, GNNs often do not obtain successful results out of the box and require extensive customization for specific tasks. This explains the existence of dozens of architectures and training procedures tailored to individual settings. Consequently, practitioners must invest significant computational resources and time into cross-validation of GNN architectures and hyperparameters to achieve competitive performance. All of this highlights the important role of *explicitly* incorporating *inductive biases* into GNN algorithms to tailor solutions to specific tasks.

This thesis introduces novel designs of inductive biases in GNNs to address practical limitations across three pivotal domains: efficient architecture search, causal inference, and inherent interpretability. The proposed models are backed with theory and/or extensive empirical evaluation. Moreover, we provide open-source implementations.

First, we propose L-CAT to address the computational complexity associated with architecture cross-validation. We show for the first time in the GNN community that learning to interpolate reduces the need for cross-validation of the GNN architectures. The proposed approach introduces minimal overhead, with only two extra learnable parameters per layer, and exhibits robustness to network initialization and input noise. These aspects make L-CAT a preferable approach in practice.

Next, we introduce VACA—a novel model based graph variational autoencoders—to perform approximate causal inference with GNNs. We first show that off-the-shelf GNN architectures are not suitable for causal inference tasks. However, we show that by leveraging the causal graph to impose constraints on the GNN architecture, causal inference queries can be approximated. As a specific application, we illustrate how VACA can be used to assess counterfactual fairness and to train a counterfactually fair classifier.

Finally, we introduce CORES to address inherently interpretable graph classification. In particular, we aim to achieve interpretability through *sparsity* in the input graph. We argue that the less information in the input given to the model, the easier the understanding of the prediction becomes. To find minimal informative subgraphs, we design a reinforcement learning pipeline that integrate conformal predictions in the design of the reward function to improve quality of the rewards and thus ease the optimization. Then, CORES jointly optimizing for sparsity and performance in a

bi-level fashion. We show empirically that CORES achieves competitive performance while using smaller input graphs compared to competing methods.

Collectively, the work presented in this thesis highlights the importance of explicit inductive biases in GNN to narrow the search space and guide optimization towards solutions that achieve competitive performance in target tasks. Our proposed GNN-based methodologies contribute to advancing the field across three domains crucial for real-world applications with potential socio-economic impact.

Zusammenfassung

Graphen sind eine leistungsstarke Datenrepräsentation, um komplexe Interaktionen zwischen Entitäten zu modellieren. Sie sind in verschiedenen Bereichen allgegenwärtig, wie z.B. in sozialen Netzwerken, der Molekularbiologie und der Verkehrsmodellierung. Message-Passing-Graph-Neural-Networks (GNNs) sind kürzlich als effektive Werkzeuge für das Lernen auf Graphen hervorgetreten und zeigen Spitzenleistungen in Anwendungen wie der Wirkstoffentdeckung, der Wettervorhersage und Empfehlungssystemen. Dennoch weisen sie sowohl theoretische als auch praktische Einschränkungen auf. Theoretische Analysen haben gezeigt, dass die Ausdruckskraft von GNN-Architekturen durch den WL-1-Isomorphism-Test beschränkt ist [99, 156], was auf eine Unfähigkeit hinweist, bestimmte Graphstrukturen zu unterscheiden. Ihre Ausdruckskraft ist auch durch Probleme wie Oversmoothing [121] und Oversquazing [5, 31] eingeschränkt, die in der Praxis die Komplexität der Architektur, beispielsweise die Anzahl der Schichten, begrenzen. Darüber hinaus zeigen theoretische und empirische Ergebnisse—einschließlich der in dieser Arbeit und früheren Arbeiten—dass die Leistung von GNNs von der jeweiligen Aufgabe und der Art des Graphen abhängt. Praktische Herausforderungen beschränken zudem die Effizienz von GNNs, einschließlich begrenzter Datenverfügbarkeit, lokaler Optima beim stochastischen Gradientenabstieg und einem riesigen Hyperparameterraum. Folglich erzielen GNNs oft nicht sofort erfolgreiche Ergebnisse und erfordern umfangreiche Anpassungen für spezifische Aufgaben. Dies erklärt die Existenz zahlreicher Architekturen und Trainingsverfahren, die auf individuelle Einstellungen zugeschnitten sind. Infolgedessen müssen Praktiker erhebliche Rechenressourcen und Zeit in die Kreuzvalidierung von GNN-Architekturen und Hyperparametern investieren, um wettbewerbsfähige Leistungen zu erzielen. All dies unterstreicht die wichtige Rolle der *expliziten* Einbindung von *induktiven Vorannahmen* in GNN-Algorithmen, um Lösungen auf spezifische Aufgaben zuzuschneiden.

Diese Dissertation stellt neuartige Designs von induktiven Vorannahmen in GNNs vor, um praktische Einschränkungen in drei zentralen Bereichen zu adressieren: effiziente Architektursuche, kausale Inferenz und inhärente Interpretierbarkeit. Die vorgeschlagenen Modelle sind theoretisch untermauert und/oder durch umfangreiche empirische Evaluierungen gestützt. Zudem bieten wir Open-Source-Implementierungen an.

Zunächst schlagen wir L-CAT vor, um die mit der Kreuzvalidierung von Architekturen verbundene Rechenkomplexität zu reduzieren. Wir zeigen erstmals in der GNN-Gemeinschaft, dass das Lernen zur Interpolation den Bedarf an Kreuzvalidierung der GNN-Architekturen reduziert. Der vorgeschlagene Ansatz führt nur zu minimalem Overhead, mit lediglich zwei zusätzlichen lernbaren Parametern pro Schicht, und zeigt Robustheit gegenüber Netzwerkinitialisierung und Eingaberauschen. Diese Aspekte machen L-CAT in der Praxis zu einem bevorzugten Ansatz.

Als nächstes führen wir VACA ein—ein neuartiges Modell basierend auf graphvariationalen Autoencodern—um eine approximative kausale Inferenz mit GNNs durchzuführen. Zunächst zeigen wir, dass herkömmliche GNN-Architekturen für kausale Inferenzaufgaben nicht geeignet sind. Wir zeigen jedoch, dass durch die Nutzung des Kausalgraphen zur Auferlegung von Beschränkungen auf die GNN-Architektur kausale Inferenzabfragen approximiert werden können. Als spezifische Anwendung illustrieren wir, wie VACA zur Bewertung von kontrafaktischer Fairness und zum Training eines kontrafaktisch fairen Klassifikators verwendet werden kann.

Schließlich stellen wir CORES vor, um eine inhärent interpretierbare Graphklassifikation zu ermöglichen. Insbesondere zielen wir darauf ab, Interpretierbarkeit durch *Sparsamkeit* im Eingabegraphen zu erreichen. Wir argumentieren, dass je weniger Informationen das Modell in der Eingabe erhält, desto einfacher wird das Verständnis der Vorhersage. Um minimale informative Teilgraphen zu finden, entwerfen wir eine Verstärkungslern-Pipeline, die konforme Vorhersagen in das Design der Belohnungsfunktion integriert, um die Qualität der Belohnungen zu verbessern und somit die Optimierung zu erleichtern. Dann optimiert CORES gleichzeitig für Sparsamkeit und Leistung in einer zweistufigen Weise. Wir zeigen empirisch, dass CORES wettbewerbsfähige Leistungen erzielt, während kleinere Eingabegraphen im Vergleich zu konkurrierenden Methoden verwendet werden.

Insgesamt unterstreicht die in dieser Dissertation vorgestellte Arbeit die Bedeutung expliziter induktiver Vorannahmen in GNNs, um den Suchraum einzugrenzen und die Optimierung hin zu Lösungen zu lenken, die in Zielaufgaben wettbewerbsfähige Leistungen erzielen. Unsere vorgeschlagenen GNN-basierten Methoden tragen zur Weiterentwicklung des Feldes in drei für reale Anwendungen entscheidenden Bereichen bei, die ein potenzielles sozioökonomisches Auswirkungen haben.

Table of Contents

Overview 1

1.1 Introduction

Over the past decade, deep learning [81] has emerged as a transformative force in artificial intelligence. This is mainly attributed to its capacity to automatically discover complex patterns and and useful representations when having access to vast amounts of data. Initially, Convolutional Neural Networks (CNNs) [82] were introduced to tackle tasks within the image domain, including but not limited to image generation [117], and image in-painting [91]. Similarly, Recurrent Neural Networks (RNNs) [54, 120] have played an extensive role in advancing temporal series tasks such as speech recognition [50] and weather forecasting [21], among others.

Another ubiquitous domain is graphs, a data structure consisting of nodes (also known as vertices) connected by edges, with inherent applications across multiple fields. For instance, in social networks, individuals and their relationships can be modeled as nodes and edges within a graph. Similarly, in biology, proteins are represented as nodes, with edges denoting their interactions. In transportation networks, cities serve as nodes, while edges depict the routes linking them.

In recent years, Graph Neural Networks (GNNs) [127], particularly those based on the message passing paradigm [25, 46], have emerged as a successful approach in graph learning, establishing themselves as the standard methodology across numerous settings [154]. They have demonstrated remarkable performance in addressing complex tasks such as link prediction in recommender systems [152], neural machine translation [12] traffic forecasting [28, 164], weather forecasting [80] and drug discovery [47, 88]. However, both theoretical analyses and empirical observations have revealed certain limitations inherent in these models.

Theoretical research has studied the expressive power of GNN architecture, focusing on their ability to discern between different graph structures. Many of these studies rely on the Weisfeiler-Lehman (WL) test of graph isomorphism framework [149] for such analysis. This test efficiently determines whether two graphs are topologically identical across a broad class of graphs [9]. Intuitively, a GNN should only assign the same embedding to two nodes if they have identical local subgraphs and matching node features. However, studies have revealed limitations in the expressivity of

GNNs: they can capture the difference between certain types of graph structures, specifically what is known as WL-1 isomorphism classes [99, 156]. This means that for some graphs that are very similar in structure, GNNs might not be able to distinguish them correctly. Additionally, they suffer other important issues such as oversquashing, characterized by the loss of information from distant nodes due to the exponential expansion of a node's receptive field resulting in numerous messages being "squashed" into fixed-size vectors [5, 31]; and oversmoothing, involving the loss of information due to repetitive feature mixing [121]. Some of these issues can be alleviated in practice, for example, by performing graph rewiring [2] or by adding noise to the optimization process [119].

Additionally, theoretical analyses (from previous work and this thesis) have shown that the performance of GNN architectures depends on the task at hand and the nature of the graphs [10, 11, 41, 59], which is challenging to know a priori. This observation is also recurrent in empirical findings [73] when comparing various GNN architectures proposed in the literature [18, 24, 53, 59, 68, 143]. In essence, there is no consistent winner that outperforms all others in every scenario, thus computationally intensive cross-validation is required.

These findings highlight the importance of explicitly introducing inductive biases—i.e., set of assumptions into the model before seeing any data—in GNN-based methodologies in order to be able to achieve high performance in practice. This is evidenced by the overwhelming number of GNN papers proposing distinct designs and methodological approaches tailored to specific tasks, as outlined in numerous surveys [15, 32, 61, 62, 72, 90, 114, 175]. Moreover, other practical challenges such as i) limited data (e.g., certain molecule classification datasets contain only a few hundred examples [26]), ii) local optima resulting from stochastic gradient descent during training, and iii) constrained computational resources limiting the number of learnable parameters and hyperparameter configurations for cross-validation, further contribute to the variability in GNN performance.

While GNNs hold promise for a wide range of tasks, it becomes apparent that the meticulous design of inductive biases for GNN architectures and optimization procedures is paramount to achieving competitive performance within resource limitations. In line with these considerations, this thesis seeks to address the following research questions:

(Q1) *How can we reduce the need for GNN architecture cross-validation?*

(Q2) *How to design GNN-based models for causal inference?*

(Q3) *How to improve inherently interpretable GNN graph classification?*

My interest in these research questions arises from the implications they can have on the deployment of GNNs Reducing the need for GNN architecture cross-validation Q 1 can speed up experimentation, enabling practitioners to allocate time to other critical tasks, such as literature review and idea generation. It also democratizes the use of GNNs by lowering the requirements for computational resources. Developing a GNN-based model for causal inference Q 2 can improve various aspects of handling interventional and counterfactual queries, including quality and inference speed. These queries are crucial for decision-making processes in fields like medicine. Finally, improving GNN interpretability Q 3 aims to increase transparency, thereby fostering greater acceptance and deployment of GNNs in real-world scenarios.

Altogether, in this thesis, we propose GNN-based methodologies backed by theory and/or extensive empirical evaluation, contributing to the advancement of the GNN field. We aim to highlight the significance and potential of careful GNN design in addressing relevant and timely topics such as efficient architecture search (Q 1), causal inference (Q 2) and inherent interpretability (Q 3). These are three aspects are especially important when the GNN-based models are used in the real-world and thus, in order to GNNs to have an impact on the economy and society.

1.2 Outline

This thesis contains three main parts, each consisting of multiple chapters. Part I provides a comprehensive overview of the background knowledge that is relevant herein. Part II offers preliminaries, an extended abstract, and future work of the main scientific contributions presented in this dissertation. Part III discusses the impact of the work, outlines promising future directions of the GNN field, and concludes with some final remarks. Additionally, Appendix A contains the complete text—both the main manuscripts and appendices—of the publications included in this thesis.

Part I contains a description of the most relevant concepts required for understanding the scientific contributions presented herein. Chapter 2 contains an introduction to graphs, that constitute the

ground data structure for our work. Section 2.2 introduces the graph theory concepts that are used in Part II. Section 2.3 outlines some of the main challenges of working with graph data. Chapter 3 introduces message-passing graph neural networks (GNNs), the foundational tool used in this thesis, and outlines their main practical limitations.

Part II tackles the primary objective of this dissertation: design inductive biases in GNNs to solve practical problems. Concretely, we address the i) computational complexity of architecture cross-validation; ii) violation of causal paths due to the message-passing scheme; and iii) black-box nature of GNNs. We tackle these challenges via the three research publications included in this thesis. Chapter 4 centers on the design of inductive biases that minimize the necessity for cross-validation of the architecture. It is based on the peer-reviewed publication Javaloy et al. [59]—presented at ICLR 2023—and aims to tackle Q 1. Chapter 5 descripts how to design a variational graph autoencoder [70] to enable approximate causal inference. This chapter, based on the peer-reviewed article Sánchez-Martín et al. [125]—presented at AAAI 2022—aims to address Q 2. Chapter 6 focuses on designing an algorithm aimed at providing interpretable and faithful predictions in graph classification by using only relevant information from the input graph. This chapter, based on Sanchez-Martin et al. [124]—preprint available at ArXiV—aims to address Q 3.

Part III synthesizes the main findings of this thesis, aligning them with the research questions posed in Section 1.1. Additionally, we present the impact of the presented work within the GNN field and suggest interesting future research directions based on the developed resarch. Finally, personal concluding remarks are provided.

1.3 List of Publications

The content presented in this thesis is mainly based on the following publications *:

Peer-reviewed publications

Pablo Sánchez-Martín*, Miriam Rateike* and Isabel Valera. "VACA: Designing Variational Graph Autoencoders for Causal Queries." Conference on Artificial Intelligence (AAAI). 2022 [125]
Paper - GitHub

Adrián Javaloy*, **Pablo Sánchez-Martín***, Amit Levi and Isabel Valera. "Learnable Graph Convolutional Attention Networks." Internacional Conference on Learning Representations (ICLR). 2023 [59]
Paper - GitHub

ArXiv publications

Pablo Sánchez-Martín, Kinaan Aamir Khan and Isabel Valera. "Improving the interpretability of GNN predictions through conformal-based graph sparsification." ArXiv. 2024 [124]
Paper - GitHub

During the course of my doctorate, I made substantial contributions to additional works that have been accepted at conferences but are not included within this thesis.

Peer-reviewed publications

Batuhan Koyuncu, **Pablo Sánchez-Martín**, Ignacio Peis, Pablo M. Olmos and Isabel Valera. "Variational Mixture of HyperGenerators for Learning Distributions Over Functions." International Conference on Machine Learning (ICML). 2023 [76]
Paper - GitHub

Adrián Javaloy, **Pablo Sánchez-Martín** and Isabel Valera. "Causal normalizing flows: from theory to practice." Conference on Neural Information Processing Systems (Oral at NeurIPS). 2023 [60]
Paper - GitHub

* Note that a superscript asterisk (*) next to the name of an author indicates equal contribution

Venues All the peer-reviewed publications introduced here have been accepted in leading conference venues in the field of artificial intelligence and machine learning, consistently ranking within the top 5 in terms of impact factor as measured by the H5 index. Sánchez-Martín et al. [125] has been accepted at the *AAAI Conference on Artificial Intelligence*, established in 1980. Javaloy et al. [59] has been accepted at the *International Conference on Learning Representations* (ICLR), founded in 1865. Koyuncu et al. [76] has been accepted at the *International Conference on Machine Learning* (ICML), established in 1981. Javaloy et al. [60] has been accepted at the *Conference on Neural Information Processing Systems* (NeurIPS) for oral presentation (awarded only to the top 0.5% of the submitted papers), inaugurated in 1987.

1.4 Personal contributions

Table 1.1 outlines my scientific contributions to each paper. The contributions are divided in generation of ideas (**Ideas**), involvement in the experimental component (**Exper.**), analysis of the results (**Analysis**), and writing (**Writing**). The contribution follow this allocation:

- Primary Contributor (PC): leading research contributions with $> 50\%$.
- Major Contributor (MC): significant shared contributions $(25\%, 50\%]$ that play a substantial role in the research.
- Co-contributor (CC): meaningful but lesser co-authorship contributions, i.e., $\leq 25\%$.

Table 1.1: Breakdown of scientific contributions. This thesis incorporates the top three papers, while the two bottom papers are excluded. The names of the proposed models are indicated in the leftmost column.

Paper	Ideas	Exper.	Analysis	Writing
VACA [125]	PC	PC	PC	MC
L-CAT [59]	MC	PC	MC	MC
CORES [124]	PC	PC	PC	PC
VaMoH [76]	MC	MC	MC	MC
CausalFlows [60]	MC	PC	MC	CC

Below we outline the contributions of the author to the papers included and excluded in this dissertation.

1.4.1 Contribution to included papers

In Sánchez-Martín et al. [125] I was first co-author, contributing the several key aspects of the manuscript. These included the definition of the core idea, development and implementation

of the proposed model, implementation of one of the baselines, creation and preparation of datasets. I also built the pipeline for conducting comparison and ablation analyses. Furthermore, I engaged in the literature review, gathering relevant previous work and synthesizing their insights. Additionally, I played a significant role in drafting the "Variational Causal Autoencoder (VACA)" and "Evaluation" sections, as well as refining the overall manuscript and some of the accompanying appendices. Moreover, I played a central role in the derivation of the proofs presented in the appendix.

In Javaloy et al. [59] I served as the first co-author, contributing significantly to all stages of manuscript development. This included shaping the idea, implementing the proposed models (CAT and L-CAT), and performing synthetic experiments. While not responsible for the theoretical framework, I reviewed all the steps in the proofs and conducted the empirical confirmation of the theoretical results. I also conducted the comparative analysis across the eleven small-scale node classification tasks.

In Sanchez-Martin et al. [124] I was the first author and made substantial contributions across all aspects of the manuscript. I played a central role in conceiving the idea and conducting the literature review. I was also the main responsible for the implementation of the proposed methodology and the creating of the experimental pipeline. Moreover, I took primary responsibility for writing the manuscript.

1.4.2 Contribution to excluded papers

In Koyuncu et al. [76], I was the second author. I initiated the project with a foundational idea and developed a prototype implementation of the proposed model. This idea and implementation was subsequently refined by the coauthors. Additionally, I contributed to the writing of the paper and conducted the experiments for the baseline models.

In Javaloy et al. [60], I was the second author. I was involved from the inception of the project. I played a significant role in all of the aspects. However, my presence was more pronounced in the experimental part, and I was less involved in the theoretical part and writing, where my focus primarily lay in crafting the appendices.

Part I

Background

Graphs 2

2.1 Introduction

Graphs are ubiquitous in many domains due to their suitability in representing complex relationships. In sociology, graphs are used to model social networks and analyze patterns of interaction between individuals [112]. Traffic prediction uses graphs to represent road networks and forecast congestion patterns [165]. Weather analysis relies on graphs to model atmospheric conditions and predict weather patterns [79]. Recommender systems use graphs to represent user-item interactions and generate personalized recommendations [43]. Also, graph-based approaches are employed in fraud detection to detect anomalous patterns and connections indicative of fraudulent activities [93]. In this chapter, our objective is to delve into the foundational principles of graph theory, providing an overview of its key concepts related to work conducted on this dissertation.

2.2 Fundamentals

A graph is formally defined as a tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, comprising a set of n nodes $\mathcal{V} = [n]$ and edges connecting node pairs $(i, j) \in \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, where i refers to the source node and j represents the destination node. Alternatively, the connectivities can be represented using the *adjacency matrix* $\boldsymbol{A} \in 0, 1^{n \times n}$, where $A_{ij} = 1$ if $(i, j) \in \mathcal{E}$, and $A_{ij} = 0$ otherwise. The collection of nodes directly linked to a node i is named the *neighborhood set* of i, denoted as $\mathcal{N}_i = \{j : (j, i) \in \mathcal{E}\}$. Each node can be associated with a feature vector $\boldsymbol{x} \in \mathbb{R}^d$ (e.g., features associated with individuals), while each edge can likewise possess a feature vector $\boldsymbol{e}_{ij} \in \mathbb{R}^{d'}$ (e.g., representing the nature of connections).

2.2.1 Types of Graphs

Graphs can be categorized depending on the presence or absence of edge directionality and weights.

Graph types based on directionality Directed graphs are characterized by directed edges, indicating one-way relationships between nodes. For instance, consider a social media network where edges represent "follows" relationships. Node i follows node j, but node j might not necessarily follow node i. Another example of a directed graph is a web page network where edges represent hyperlinks. If page i has a hyperlink pointing to page j, it does not necessarily imply that page j has a hyperlink pointing back to page i. In contrast, undirected graphs lack edge directionality, indicating symmetric relationships. An example could be a friendship network, where edges represent mutual friendships. If node i is friends with node j, then node j is also friends with node i. Another example of an undirected graph could be a co-authorship network, where edges represent collaborations between researchers.

Type of graphs based on weighted edges Graphs can be classified into weighted and unweighted based on the presence of numerical values associated with edges. In weighted graphs, edges are assigned numerical values representing the strength or cost of the relationship between nodes. For example, in a transportation network, the weight of an edge could represent the distance between two cities or the time it takes to travel between them. Conversely, unweighted graphs have edges with no associated weight, denoted by binary values. For instance, in a social network where edges represent friendships, the absence or presence of an edge between two individuals signifies either no friendship ($A_{ij} = 0$) or friendship ($A_{ij} = 1$), respectively.

2.2.2 Important Properties

There exist numerous quantities that can be computed on graphs which collectively characterize them and help in their analysis. Herein, we outline the most relevant metrics for the research conducted in this dissertation.

Size The size of a grap corresponds to the total number of nodes ($|\mathcal{V}|$) or edges ($|\mathcal{E}|$) in the graph.

Density The graph density quantifies the ratio of actual edges to the total possible edges within the graph. For undirected graphs it is computed as:

$$\delta = \frac{2 \times |\mathcal{E}|}{|\mathcal{V}|\,(|\mathcal{V}| - 1)}, \tag{2.1}$$

and for directed graphs it is computed as:

$$\delta = \frac{|\mathcal{E}|}{|\mathcal{V}|\,(|\mathcal{V}| - 1)}. \tag{2.2}$$

Node Degree The node degree is the number of edges incoming (in-degree) or outgoing (out-degree) to a node. For a node $v \in V$, its in-degree $d_i(v)$ is calculated as:

$$d_i(v) = |\{u \in \mathcal{V} : (u, v) \in \mathcal{E}\}|, \tag{2.3}$$

and its out-degree $d_o(v)$ is calculated as:

$$d_o(v) = |\{u \in \mathcal{V} : (v, u) \in \mathcal{E}\}|. \tag{2.4}$$

Note that for undirected graphs we simply compute the node degree d_n.

Paths and Cyles A path $\mathcal{P} = (e_1, \ldots, e_k)$ of size k in a directed graph is a sequence of of edges $e_i = (u_{i-1}, u_i)$ directed in the same direction such that all the nodes are distinct. A cycle of size k is a path that has $u_1 = u_k$. The analysis of paths and cycles within a graph offers insights into its structural properties and connectivity patterns.

Connected components A connected component of a graph is a set of vertices that are linked to each other by paths.

Shortest path between nodes The shortest path between two nodes u and v is denoted as $\mathcal{P}_{uv}$ and refers to the path between u and v with the minimal number of edges.

Distance between nodes The distance $d_{uv} = |\mathcal{P}_{uv}|$ between two nodes u and v is size of the shortest path.

Eccentricity The eccentricity $\epsilon(v)$ of a node v is the greatest distance between v and any other node in the graph:

$$\epsilon(v) = \max_{u \in \mathcal{V}} d_{vu}. \tag{2.5}$$

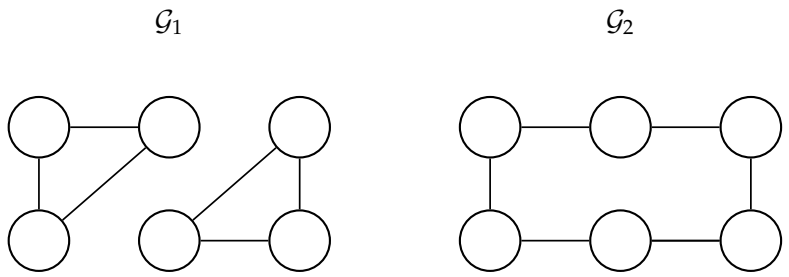


Figure 2.1: Example of two graphs indistinguishable by the WL-1 test.

Diameter The diameter of a graph, denoted as d, represents the maximum distance between any pair of nodes within the graph. It is computed as:

$$d = \max_{u \in \mathcal{V}} \epsilon(u). \tag{2.6}$$

The diameter computation involves initially determining the shortest path between each pair of nodes and subsequently identifying the path with the longest length, thus indicating the diameter of the graph.

2.2.3 Graph Isomorphism

Graph isomorphism is the task of determining whether two graphs have identical structures despite differences in node labels or ordering. In the field of graph theory, two graphs $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$ are said to be isomorphic if there exists a bijection (a one-to-one correspondence) between their node sets $\mathcal{V}_1$ and $\mathcal{V}_2$, denoted as

$$\mathbf{f} : \mathcal{V}_1 \mapsto \mathcal{V}_2, \tag{2.7}$$

where for any pair of nodes $u, v \in \mathcal{V}_1$, u and v are adjacent in $\mathcal{E}_1$ if and only if $\mathbf{f}(u)$ and $\mathbf{f}(v)$ are adjacent in $\mathcal{E}_2$. In essence, the structure of the graphs remains unchanged under the bijection. This preservation can be expressed symbolically as:

$$\forall u, v \in \mathcal{V}_1, (u, v) \in \mathcal{E}_1 \iff (\mathbf{f}(u), \mathbf{f}(v)) \in \mathcal{E}_2. \tag{2.8}$$

The task of testing for the existence of isomorphism between two graphs is known to be computationally challenging, falling within the family of NP-hard problems. However, heuristics offer practical solutions. One popular heuristic is the Wiesfeiler-Leman (WL) graph isomorphism test [84]. The WL test iteratively refines node labels using the local graph structures. It operates under the assumption that isomorphic graphs should exhibit similar

"neighborhood subgraphs" after several iterations of the labeling process. In other words, isomorphic graphs should produce the same sets of labels for their nodes. In particular, the 1-WL test considers the 1-hop neighborhoods of nodes, refining their labels based on the labels of their immediate neighbors, and iteratively checks if two graphs can be distinguished by comparing these refined labels. The algorithm runs in polynomial time, making it practical for comparing large-scale graphs, and it is widely use to analyze the expressive power of GNN architectures. However, the 1-WL test does not guarantee to always produce the correct results. For example, the non-isomorphic graphs shown in Figure 2.1 are "1-WL indistinguishable" graphs because their nodes have identical connectivity (i.e., two neighbors), producing identical labels through all iterations. Consequently, the 1-WL test may yield false positives, erroneously identifying non-isomorphic graphs as isomorphic.

2.2.4 Operations on Graphs

Generally speaking, graph operations typically fall into two categories: permutation invariant and permutation equivariant operations. Let us consider a function $\mathbf{f}$ that takes n inputs $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}$.

Permutation invariance A function $\mathbf{f}$ is permutation invariant if, for any permutation π, it satisfies:

$$\mathbf{f}(\boldsymbol{x}_\pi) = \mathbf{f}(\boldsymbol{x}). \tag{2.9}$$

In essence, rearranging the inputs in any order does not changes the output of the operation. Examples of such functions include summation, maximum, and mean operations.

Permutation equivariance A function $\mathbf{f}$ is permutation equivariant if, for any permutation π, it holds that:

$$\mathbf{f}(\boldsymbol{x}_\pi) = \mathbf{f}(\boldsymbol{x})_\pi. \tag{2.10}$$

In simpler terms, rearranging the inputs in any order results in the output of the operation being rearranged in the same manner. An

example of such a function is the graph Laplacian matrix $\boldsymbol{L}$ which is defined as:

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}, \quad \text{where} \quad D_{ij} = \begin{cases} d_i & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{2.11}$$

The matrix $\boldsymbol{D}$ is known as the degree matrix of the graph given by $\boldsymbol{A}$.

2.2.5 Tasks on Graphs

Tasks on graphs can be classified based on the available information during training (a.k.a. learning process) and the scope of predictions.

Availability of information Tasks are distinguished as transductive or inductive depending on the information accessible during training. Transductive tasks focus on predicting labels or properties for data points within a given graph, using information solely from the provided graph. This approach aims to predict exclusively for the data points present in the training set without extending to unseen data points. Conversely, inductive tasks focus on extrapolating patterns and relationships learned from the training data to predict for unseen data points, potentially on entirely novel graphs. Inductive learning aims to make predictions beyond the scope of the training set.

Scope of prediction Tasks are further categorized into graph, node, and edge-level tasks. Graph-level tasks involve predicting characteristics of the entire graph and are typically inductive. Node-level tasks may be inductive or transductive, focusing on predicting attributes or labels of individual nodes. Similarly, edge-level tasks can also be inductive or transductive, targeting predictions concerning relationships between pairs of nodes.

2.3 Challenges in graph learning

Graph learning poses several challenges due to the inherent characteristics of graphs and the complexity of real-world applications.

Irregular structure The irregular structure of graphs, such as unfixed grids, poses challenges for standard neural networks designed for regular data structures like tabular data, images or sequences.

Scalability Real-world graphs, such as social networks, can contain millions or even billions of nodes, making scalability a significant challenge for graph learning algorithms.

Heterogeneous graphs Real-world graphs often exhibit heterogeneity, involving different types of node and edge information. For example, in social networks where nodes represent users, features may include both continuous (e.g., income) and discrete (e.g., place of birth) attributes. Similarly, edges may represent various types of connections such as friendship, work collaboration, etc.

Data Sparsity Graphs are typically sparse (i.e., have low density δ), meaning that only a small fraction of all possible edges are present. This sparsity poses challenges for learning algorithms, especially when distant nodes in the graph may be relevant for solving certain tasks. Graphs exhibiting such characteristics are known as heterophilous graphs, where linked nodes are prone to have different labels or dissimilar features. For instance, in social network graphs, automated bots tend to establish connections with users rather than with other bots [49].

3 Graph Neural Networks

This chapter provides an introduction to message-passing graph neural networks (GNNs). Section 3.1 describes the general formulation for a GNN layer. Section 3.2 explores the most prominent families of GNN architectures. Finally, Section 3.3 introduces the most important limitations inherent in GNNs.

3.1 Introduction

Message-passing Graph Neural Networks (GNNs) [25, 46] are designed for graph-structured data processing. Let us consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $n = |\mathcal{V}|$. Each node $v \in \mathcal{V}$ has a feature vector $\boldsymbol{h} = \boldsymbol{h}^0$, and a GNN layer transforms these features using neighborhood information. Formally, a GNN performs L update rounds, updating each node' s feature vector $\boldsymbol{h}_i^l$ at step $1 \leq l \leq L$:

$$\boldsymbol{h}_i^l = \mathbf{f}_{\theta_u}\left(\boldsymbol{h}_i^{l\text{-}1}, \bigoplus_{j \in \mathcal{N}} \gamma_{ij} \mathbf{f}_{\theta_m}(\boldsymbol{h}_i^{l\text{-}1}, \boldsymbol{h}_j^{l\text{-}1})\right). \tag{3.1}$$

Here, $\mathbf{f}_{\theta_u}$ and $\mathbf{f}_{\theta_m}$ are differentiable, parameterized functions (e.g., mutilayer perceptrons), $\mathcal{N}_i$ represents node i's neighbors, $\bigoplus$ denotes permutation invariant operations (i.e., sum, mean, or max), and γ_{ij} is the attention coefficient of the message from node j to node i. First, node i receives a message $\mathbf{f}_{\theta_m}(\boldsymbol{h}_i^{l\text{-}1}, \boldsymbol{h}_j^{l\text{-}1})$ from each of its neighbors $j \in \mathcal{N}_i$. Then, these messages are aggregated via $\bigoplus$. Finally, $\boldsymbol{h}_i^l$ is computed as a function $\mathbf{f}_{\theta_u}$ of the node's previous state $\boldsymbol{h}_i^{l\text{-}1}$ and the aggregated message. After L steps, we obtain the final node representations $\boldsymbol{h}_i^L \forall i \in \mathcal{V}$. For graph-level prediction tasks, a permutation-invariant readout function $\boldsymbol{z} = \textsc{ReadOut}\left(\{\boldsymbol{h}_i^L : i \in \mathcal{V}\}\right)$ is employed that ensures the output is node order independent. For a more complete description of other modules composing a GNN layers (e.g., dropout, pooling, batch normalization) please refer to Zhou et al. [178].

Definition 3.1 (Receptive field from Topping et al. [141]) Let $\mathcal{G}$ be an undirected, connected graph and $\mathcal{P}_{ij}$ be the shortest path between nodes i and j. Then we define the receptive field $B_r(i)$ of node i of an L-layer GNN as

$$B_r(i) = \{j \in \mathcal{V} : \left|\mathcal{P}_{ij}\right| \leq L\}. \tag{3.2}$$

Basically, this indicates that a GNN comprising L layers imposes constraints on the flow of information to each node. Specifically, node i can only receive information from node j through paths with lengths $\left|\mathcal{P}_{ji}\right| \leq L$. This implies that at the final layer of a GNN, for a node i information from node j is accessible if and only if the shortest path between them has a length less than or equal to the number of GNN layers L. Additionally, the longest the shortest distance between two nodes is, the less information will generally flow between them. Therefore, when designing a GNN architecture, it is crucial to take into account the dependencies among nodes required by the task at hand. This consideration is particularly significant in the development of VACA in Chapter 5.

3.2 Types of architectures

There are many different types of architectures proposed in the literature, each of them is mainly characterized by the way that we define the different components of Equation (3.1). Depending on the way the coefficients γ_{ij} (see Equation (3.1)) are computed, we identify different GNN architectures.

Graph convolutional networks (GCNs) [68] are simple yet effective. In short, GCNs define the message as $\mathbf{f}_{\theta_m}(\boldsymbol{h}_i^{l\text{-}1}, \boldsymbol{h}_j^{l\text{-}1}) = \boldsymbol{h}_j^{l\text{-}1}$ and compute the average of the messages, i.e., they assign the same coefficient $\gamma_{ij} = 1/|\mathcal{N}_i^\star|$ to every neighbor:

$$\boldsymbol{h}_i^l = \mathbf{f}_{\theta_u}\left(\frac{1}{|\mathcal{N}_i^\star|}\sum_{j\in\mathcal{N}_i^\star} \boldsymbol{W}_v \boldsymbol{h}_j^{l\text{-}1}\right), \tag{3.3}$$

where $\mathcal{N}_i^\star = \mathcal{N}_i \cup \{i\}$.

Graph attention networks take a different approach. Instead of assigning a fixed value to each coefficient γ_{ij}, they compute it as a function of the sender and receiver nodes. A general formulation

for these models can be written as follows:

$$\boldsymbol{h}_i^l = \mathbf{f}_{\theta_u}\left(\frac{1}{|\mathcal{N}_i^\star|}\sum_{j\in\mathcal{N}_i^\star}\gamma_{ij}\boldsymbol{W}_v\boldsymbol{h}_j^{l\text{-}1}\right) \quad \gamma_{ij} = \frac{\exp(\Psi(\boldsymbol{h}_i,\boldsymbol{h}_j))}{\sum_{\ell\in\mathcal{N}_i^*}\exp(\Psi(\boldsymbol{h}_i,\boldsymbol{h}_\ell))} \tag{3.4}$$

Here, $\Psi(\boldsymbol{h}_i,\boldsymbol{h}_j) = \alpha(\boldsymbol{W}_q\boldsymbol{h}_i,\boldsymbol{W}_k\boldsymbol{h}_j)$ is known as the *score function* (or *attention architecture*), and provides a score value between the messages $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ (or more generally, between a learnable mapping of the messages). From these scores, the (attention) coefficients are obtained by normalizing them, such that $\sum_j \gamma_{ij} = 1$. We can find in the literature different attention layers which differ in the way they define the score function. Throughout this work, particularly in Chapter 4, we focus on the original GAT [143] and its extension GATv2 [18]:

$$\text{GAT:} \quad \Psi(\boldsymbol{h}_i,\boldsymbol{h}_j) = \text{LeakyRelu}\left(\boldsymbol{a}^\top[\boldsymbol{W}_q\boldsymbol{h}_i||\boldsymbol{W}_k\boldsymbol{h}_j]\right) , \tag{3.5}$$

$$\text{GATv2:} \quad \Psi(\boldsymbol{h}_i,\boldsymbol{h}_j) = \boldsymbol{a}^\top\text{LeakyRelu}\left(\boldsymbol{W}_q\boldsymbol{h}_i + \boldsymbol{W}_k\boldsymbol{h}_j\right) , \tag{3.6}$$

where the learnable parameters are now the attention vector $\boldsymbol{a}$; and the matrices $\boldsymbol{W}_q$, $\boldsymbol{W}_k$, and $\boldsymbol{W}_v$. Following previous work [18, 143], we assume that these matrices are coupled, i.e., $\boldsymbol{W}_q = \boldsymbol{W}_k = \boldsymbol{W}_v$. Note that the difference between the two layers lies in the position of the vector $\boldsymbol{a}$. Brody et al. [18] showed that by taking $\boldsymbol{a}$ out of the nonlinearity increases the expressiveness since the product of $\boldsymbol{a}$ and a weight matrix does not collapse into another vector.

3.3 Limitations

GNNs have emerged as powerful tools for learning representations from graph-structured data. However, despite their effectiveness, GNNs are not without limitations. In this section, we discuss tho of the main limitations: oversmoothing and oversquashing.

Definition 3.2 (Oversmoothing adapted from Rusch et al. [121]) Let $\mathcal{G}$ be an undirected, connected graph and $\boldsymbol{H}^l \in \mathbb{R}^{n\times d}$ denote the l-th layer hidden features of a L-layer GNN defined on $\mathcal{G}$. Moreover, we call $\mu : \mathbb{R}^{n\times d} \mapsto \mathbb{R}_{\geq 0}$ a **node-similarity measure** if it satisfies the following axioms:

1. $\exists \boldsymbol{c} \in \mathbb{R}^d$ with $\boldsymbol{h}_i^l = \boldsymbol{c}$ for all nodes $i \in \mathcal{V} \iff \mu\left(\boldsymbol{H}^l\right) = 0$, for $\boldsymbol{H}^l \in \mathbb{R}^{n\times d}$.
2. $\mu\left(\boldsymbol{X}+\boldsymbol{Y}\right) \leq \mu\left(\boldsymbol{X}\right) + \mu\left(\boldsymbol{Y}\right)$, for all $\boldsymbol{X},\boldsymbol{Y} \in \mathbb{R}^{n\times d}$.

We then measure **over-smoothing with respect to** μ as the layewise exponential convergence of the node-similarity measure μ to zero, i.e.,

3. $\mu\left(\boldsymbol{H}^l\right) \leq C_1 e^{-C_2 l}$, for $l = 0, \ldots, L$ with some constants $C_1, C_2 > 0$.

In simple terms, oversmoothing refers to the phenomenon where the *node representations in a GNN become indistinguishable from each other after multiple layers of aggregation* [154]: $\mu\left(\boldsymbol{H}^l\right) \to 0$ as l increases. As indicated in Condition 3 in Definition 3.2, the *issue increases expontentially* as we evaluate μ on *deeper layers*. This indicates that excessive information propagation can lead to loss of discriminative power in the learned representations. Some methods for reducing oversmoothing include normalization and regularization, such as node-wise normalization of the feature vectors [179] or randomly dropping edges during traning [119]. Other methods directly modify the dynamics of the message-passing propagation [122] or add residual connections [22].

Definition 3.3 (Oversquashing adapted from Di Giovanni et al. [31]) Let $\mathcal{G}$ be an undirected, connected graph and $\boldsymbol{h}_i^l \in \mathbb{R}^d$ denote the hidden feature at the l-th layer for node i in a L-layer GNN defined on $\mathcal{G}$. Let us also consider two nodes $i, j \in \mathcal{V}$ separated by distance $d_{ij} = r$. We then measure **oversquashing** by the exponential decay of the norm of the derivative of $\boldsymbol{h}_i^r$ with respect to $\boldsymbol{h}_j^0$, which is upper bounded by:

$$\left\| \frac{\partial \boldsymbol{h}_i^r}{\partial \boldsymbol{h}_j^0} \right\|_{L_1} \leq c^r \left(\tilde{\boldsymbol{A}}^r\right)_{ij}. \tag{3.7}$$

A small derivative of $\boldsymbol{h}_i^r$ with respect to $\boldsymbol{h}_j^0$ indicates that after r layers, the feature at node i is mostly insensitive to the information initially contained at node j, suggesting ineffective message propagation. This decay is attributed to two primary factors: a *model-dependent* constant c, influenced by the architectural choices of the GNN (e.g., feature dimensionality d and choice of the non-linear activation functions), and *graph topology* factor $\tilde{\boldsymbol{A}}$, which represents a function of the adjacency matrix $\boldsymbol{A}$. The latter factor is important. Note that the power of the adjacency matrix represents the count of paths of a certain length between pairs of nodes in a graph, which decreases exponentially with the exponent. This means that oversquashing happens at an exponential rate with the distance between the nodes in the graph. Remarkably, it happens independently of the GNN model. Strategies to mitigate oversquashing often involve modifying the structure of the graph, e.g., through graph rewiring, to improve the interconnectivity of

nodes [31]. However, such modifications (e.g. adding non-existent links) can distort the inherent meaning of the graph, particularly in domains where graph topology has precise functionality, such as molecular structures.

The identified limitations of oversmoothing and oversquashing pose significant challenges for the successful application of GNNs. In particular, in causal inference, when information transfer between nodes is compromised interventions on specific graph regions (e.g., through graph surgery) may have negligible impact. This scenario undermines the capacity of GNNs to accurately model effects of such interventions since the GNN fails to encapsulate the true causal dependencies of the system. In Chapter 5, we elaborate more on this issue.

Part II

Mitigating Practical Limitations of GNNs through Explicit Inductive Biases

4 Learnable Graph Convolutional Attention Networks

In this chapter, we present our work Javaloy et al. [59] whose objective is to reduce the need for cross-validation of the GNN architectures, thus tackling research question Q 1. Full paper can be found in Appendix A.

4.1 Extended Abstract

Existing Graph Neural Networks (GNNs) compute the message exchange between nodes by either aggregating uniformly (*convolving*) the features of all the neighboring nodes, or by applying a non-uniform score (*paying attention*) to the features. This results in two very broad families of GNNs, namely GCNs [68] and GATs [143]. Previous works have shown the strengths and limitations of both approaches from a theoretical [10, 11, 41], and empirical [73] point of view. These works showed scenarios for which GCNs can be beneficial in the absence of noise, and that GAT can outperform GCNs in other scenarios, leaving open the question of which architecture is preferable in terms of performance. In short, these works have shown that their performance depends on the nature of the data at hand (i.e., the graph and the features). As a consequence, the standard approach in practice is to select between GCNs and GATs via computationally demanding cross-validation. Moreover, the type of layer is usually shared along the GNN, as selecting an architecture for each layer in the GNN results in a combinatorial number of choices.

In Javaloy et al. [59], we aim to exploit the benefits of both convolution and attention operations in the design of GNN architectures. To this end, we first introduce a novel graph convolutional attention layer (CAT), which extends existing attention layers by taking the convolved features as inputs of the score function:

$$\Psi(\boldsymbol{h}_i, \boldsymbol{h}_j) = \alpha(\boldsymbol{W}\tilde{\boldsymbol{h}}_i, \boldsymbol{W}\tilde{\boldsymbol{h}}_j) \quad \text{where} \quad \tilde{\boldsymbol{h}}_i = \frac{1}{|N_i^*|} \sum_{\ell \in N_i^*} \boldsymbol{h}_\ell \,. \tag{4.1}$$

Following Fountoulakis et al. [41], we propose a graph generative model $\mathcal{G} \sim \text{CSBM}(n, p, q, \boldsymbol{\mu}, \sigma^2)$—based on contextual stochastic block model [29]—to theoretically compare GCN, GAT, and CAT architectures (i.e., GNNs architectures using either convolution,

attention, or convolutional attention operations in all their GNN layers).

Importantly, the parameters $\boldsymbol{\mu} \in \mathbb{R}^d_{\geq 0}$ and $0 < q < 1$ change the difficulty of the problem with respect to the node features and graph topology, respectively. Our theoretical analysis shows that, unfortunately, there is *no free lunch* among these GNN architectures. Their performance is fully data-dependent, i.e., depends on the graph and features of the nodes. Figure 4.1 empirically validates our theoretical results on a node classification task using the synthetic data sampled from the generative model introduced in the previous paragraph. The top row plots show the accuracy for hard (left, $\|\boldsymbol{\mu}\| = 0.1$) and easy (right, $\|\boldsymbol{\mu}\| = 4.3$) regimes based on node features $\boldsymbol{\mu}$ as q varies. In the hard regime, GAT (pink line) fails to achieve perfect performance, whereas CAT (blue line) achieves perfect performance when q is sufficiently small. The bottom row plots show the accuracy for hard (left, $q = 0.3$) and easy (right, $q = 0.1$) regimes based on graph topology q as $\|\boldsymbol{\mu}\|$ varies. We observe a transition in the accuracy of both GAT and CAT as a function of $\|\boldsymbol{\mu}\|$. Depending on $\|\boldsymbol{\mu}\|$, either GAT or CAT is preferable. When $\|\boldsymbol{\mu}\|$ is large enough, both architectures achieve perfect performance. These experimental results corroborate our theoretical findings that *no free lunch* exists among GCN, GAT, and CAT architectures.

We argue that this issue can be easily overcome by learning to interpolate between the three. First, note that GCN (see Equation (3.3)) and GAT (see Equation (3.4)) only differ in that GCN weighs all neighbors equally and GAT weights them using the coefficients γ_{ij}. Also note that the more similar the attention scores $\Psi(\boldsymbol{h}_i, \boldsymbol{h}_j)$ are, the more uniform the coefficients γ_{ij} are. Thus, we can interpolate between GCN and GAT by introducing a learnable parameter, namely $\lambda_1 \in [0, 1]$. Similarly, the formulation of GAT and CAT differ in the convolution within the score, which can be interpolated with another learnable parameter, namely $\lambda_2 \in [0, 1]$.

Following these observations, we propose the *learnable convolutional attention layer* (L-CAT), which can be formulated as an attention layer with the following score:

$$\Psi(\boldsymbol{h}_i, \boldsymbol{h}_j) = \lambda_1 \cdot \alpha(\boldsymbol{W}\tilde{\boldsymbol{h}}_i, \boldsymbol{W}\tilde{\boldsymbol{h}}_j) \quad \text{where} \quad \tilde{\boldsymbol{h}}_i = \frac{\boldsymbol{h}_i + \lambda_2 \sum_{\ell \in \mathcal{N}_i} \boldsymbol{h}_\ell}{1 + \lambda_2 |\mathcal{N}_i|}. \tag{4.2}$$

where $\lambda_1, \lambda_2 \in [0, 1]$ are two learnable parameters. This formulation lets L-CAT learn to interpolate between GCN ($\lambda_1 = 0$), GAT ($\lambda_1 = 1$ and $\lambda_2 = 0$), and CAT ($\lambda_1 = 1$ and $\lambda_2 = 1$) and enables a number of non-trivial benefits. Not only can it switch between

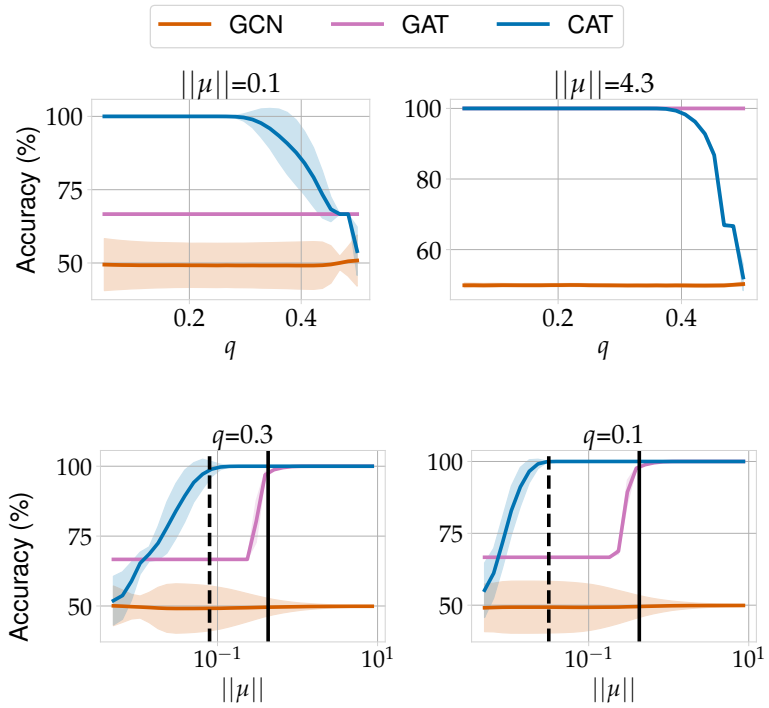
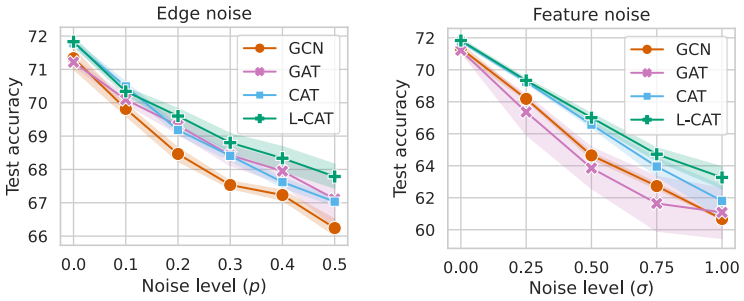


Figure 4.1: Empirical validation of theoretical results using synthetic data. The top-row plots show accuracy as we vary the noise level q for $\|\boldsymbol{\mu}\| = 0.1$ and $\|\boldsymbol{\mu}\| = 4.3$. The bottom-row plots show the accuracy as we change the norm of the means $\|\boldsymbol{\mu}\|$ for $q = 0.1$ and $q = 0.3$. We use two vertical lines to present the classification threshold stated in the theorems of the paper, for GAT (solid line) and for CAT (dashed line). We observe that these is not family of architectures that achieves the best performance in all the scenarios.

existing layers, but it also learns the amount of attention necessary for each use-case. Moreover, by comprising the three layers in a single learnable formulation, it removes the necessity of cross-validating the type of layer, as their performance is data-dependent, as mentioned before. Remarkably, it allows to easily combine different layer types within the same architecture, a process that was prohibitively expensive prior to this work.

We conduct a thorough empirical analysis, that can be reproduced using the code at `https://github.com/psanch21/LCAT`. Firstly, we validate our theoretical findings on synthetic data. Secondly, we show through 11 small-scale node classification tasks that L-CAT is as competitive as the baseline models. Then, we move to more demanding scenarios from the Open Graph Benchmark [55], demonstrating that L-CAT is a more flexible and robust alternative to its baseline methods that reduces the need for cross-validating without giving up on performance. Finally, we analyze the capabilities of L-CAT considering two important aspects for real-world applications. First, we explore the robustness of the proposed models to different levels of noise, i.e., we attempt to simulate scenarios where there exist measurement inaccuracies in the input features and edges. Figure 4.2 summarizes the results. We can observe that L-CAT (green line) consistently outperforms the other architectures across all noise levels (x-axis) for both types of noise. Second, we explore the robustness of L-CAT to network initialization, i.e., the ability to obtain satisfying performance independently of the initial parameters. We show that L-CAT consistently obtains high



(a) $\mathcal{E} = \mathcal{E} \cup \mathcal{E}_p$

(b) $\boldsymbol{x} = \boldsymbol{x} + \mathcal{N}(\mathbf{0}, \mathbf{I}\sigma)$

Figure 4.2: Robustness to noise results. Figure 4.2a shows the results adding random edges $\mathcal{E}_p$ with probability p. Figure 4.2b shows the results with noise level σ added to node features.

accuracy in all scenarios and runs, which in turn reduces the need of cross-validating the initialization strategy. As a result, L-CAT proved to be a viable drop-in replacement that removes the need to cross-validate the layer type.

4.2 Discussion

During the last years, the field of GNNs has experienced a drastic increase in the number of GNN architectures proposed, nowadays ranging in the dozens [154]. In practice, this means that whenever we face a new problem we aim to tackle with a GNN, it is computationally unfeasible to try all the available GNN architectures thus the practitioner has to make a selection of those that seem more promising, which in many cases simplifies to those that are more familiar or more famous.

In Javaloy et al. [59], we studied how to combine the strengths of two of the most famous families of architectures, convolution [68] and attention [143] GNN layers, aiming at reducing the amount of architecture cross-validation. First, we introduced CAT, which computes attention with respect to the convolved features, and analyzed its advantages and limitations on a new synthetic dataset. This analysis revealed different regimes where one model is preferred over the others, reinforcing the idea that selecting between GCNs, GATs, and now CATs is a difficult task, as their performance depends directly on the data. For this reason, we proposed L-CAT, a model that interpolates between the three via two learnable parameters (per layer). We showed the effectiveness of L-CAT with extensive experimental results, yielding competitive performance while being more robust than other architectures.

As a result, L-CAT proved to be a viable drop-in replacement that removes the need to cross-validate the layer type.

Future work

Exploring the application of L-CAT to other GNN families beyond GCN and GIN is an interesting direction for research. Since L-CAT simply interpolates between two different adjacency matrices, it is easily adaptable to other GNN architectures, such as Principal Neighborhood Aggregation (PNA) [24] and Graph Convolutional Network II (GCNII) [23]. We are confident that the incorporation of learnable interpolation mechanisms can enhance performance and, in particular, reduce the need for expensive and extensive cross-validation. We hope that our work on L-CAT will inspire new and exciting research.

VACA: Designing Variational Graph Autoencoders for Causal Queries

5

In this chapter, we present our work Sánchez-Martín et al. [125], where our goal is to carefully design a variational graph autoencoder capable of performing causal inference. Specifically, we aim to answer interventional and counterfactual queries, thus to answer research question Q 2.

5.1 Preliminaries

In this section, we introduce the related concepts important for the understanding of the chapter. Firstly, we offer an introduction to causal inference, specifically within the framework of the structural causal model (SCM) [110, 111]. Our focus lies on defining an SCM and introducing the distributions we can model with the framework, namely observational, interventional, and counterfactual distributions. Subsequently, we introduce the variational graph autoencoder, the foundational model used in constructing VACA.

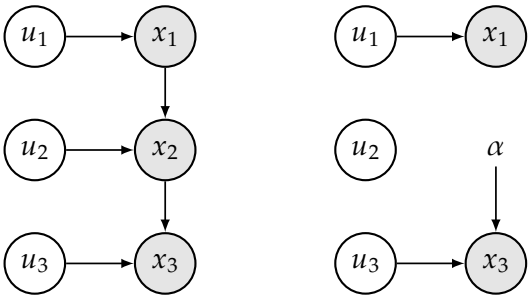
5.1.1 Causal inference

Causal inference refers to the process of identifying and quantifying causal relationships among observed variables. It involves the study of methods to address interventional questions, such as *How will doubling the dose of medication affect the patient's blood pressure?*, and counterfactual questions, such as*If I had bought that beach house instead of that suburban house, how would my lifestyle have changed?*. A popular framework for performing causal inference is the structural causal model (SCM) [110, 111].

Structural Causal Model

A SCM is a tuple $\mathcal{M} = (\tilde{\mathbf{f}}, P_{\mathbf{u}})$ describing a data-generating process that transforms a set of d exogenous random variables, $\mathbf{u} \sim P_{\mathbf{u}}$, into a set of d (observed) endogenous random variables, $\mathbf{x}$, through the *structural equations* $\tilde{\mathbf{f}} = \{\tilde{f}_i\}_{i=1}^{d}$. Specifically, the endogenous variables are computed as follows:

$$\mathbf{u} := (\mathrm{u}_1, \mathrm{u}_2, \dots, \mathrm{u}_d) \sim P_{\mathbf{u}}\,, \quad \mathrm{x}_i = \tilde{f}_i(\mathbf{x}_{\mathrm{pa}_i}, \mathrm{u}_i)\,, \quad \text{for } i = 1, 2, \dots, d\,. \tag{5.1}$$



(a) *chain* SCM **(b)** Intervened *chain* SCM

Figure 5.1: *chain* SCM. Figure 5.1a illustrates a *chain* SCM with three endogenous variables $\mathbf{x}$ and fully factorized exogenous variable distribution $p(\mathbf{u}) = \prod_i p(\mathrm{u}_i)$ (causal sufficiency). Figure 5.1b depicts the same SCM under the intervention $do(\mathrm{x}_2 = \alpha)$.

In essence, each i component of $\tilde{\mathbf{f}}$ maps the i exogenous variable u_i to the i endogenous variable x_i, given the subset of endogenous variables directly causing x_i, i.e., the causal parents $\mathbf{x}_{\mathrm{pa}_i}$. Here, the exogenous variables represent factors external to the model, whose causes are not considered within the scope of the model. A common assumption is *causal sufficiency*, where the exogenous distribution is factorized over the dimensions $P_{\mathbf{u}} = \prod_i p(u_i)$.

An SCM also induces a *causal graph* $\mathcal{G} := (\mathcal{V}, \mathcal{E})$, a graphical representation of the causal model. $\mathcal{V} \in [d]$ represents the set of endogenous and exogenous variables, and $(i, j) \in \mathcal{E}$ indicates the causal parent-child relationship between variables [110]. In this work, we assume the causal graph to be a Directed Acyclic Graph (DAG). Furthermore, we define the *ancestors* of x_i (denoted as an_i) as the direct and indirect causes. Refer to Figure 5.1a for an illustration of the causal graph of a *chain*, a specific SCM where $\mathrm{x}_1 = \tilde{f}_i(\mathrm{u}_1)$, $\mathrm{x}_2 = \tilde{f}_i(\mathrm{x}_1, \mathrm{u}_2)$, and $\mathrm{x}_3 = \tilde{f}_i(\mathrm{x}_2, \mathrm{u}_3)$. Here, for instance, x_1 is a parent of x_2, and $\mathrm{an}_2 = \{\mathrm{u}_2, \mathrm{x}_1, \mathrm{u}_1\}$. The causal graph aids in visually identifying (conditional) independence relations between variables, although it does not contain information about the mechanism of causation. In addition, an SCM induces a joint distribution over the endogenous variables $p(\mathbf{x}) = \prod_i p(\mathrm{x}_i | \mathbf{x}_{\mathrm{pa}_i}, \mathrm{u}_i)$, on which we want to answer causal queries.

Interventional and counterfactual queries

Given an SCM, there are two types of causal queries of general interest: interventional queries, e.g., "What would happen to the population $\mathbb{X}$, if variable x_i would be set to a fixed value α?", and counterfactual queries, e.g.,"What would have happened to a specific factual sample $\mathbf{x}^{\mathrm{f}} \in \mathbb{X}$, had x_i been set to a value α?".

Interventional queries A *interventional query* aims to evaluate the effect at the population level of a specific intervention on, or equivalently manipulations of, a subset of the endogenous

variables $\mathcal{I} \subseteq [d] := \{1, \ldots, d\}$. Interventions on an SCM $\mathcal{M}$ are often represented with the *do-operator* $do(\mathrm{x}_i = \alpha)$ [111] and lead to a modified SCM $\mathcal{M}^{\mathcal{I}}$ which induces a new distribution over the set of endogenous variables $p(\mathbf{x} \mid do(\mathrm{x}_i = \alpha))$, which is referred to as the *interventional distribution*. In $\mathcal{G}$ an intervention removes incoming edges to node i and sets $\mathrm{x}_i = \alpha$, see Figure 5.1b for an illustration with the *chain* SCM. This is usually refered to as perform *graph surgery*.

Intervention versus conditioning When we intervene, we change the system, and the values of other variables often change as a result. When we condition on a variable, we change nothing; we merely narrow our focus to the subset of cases in which the variable takes the value we are interested in. To further illustrate, consider a medical scenario involving a doctor and patient treatments:

- **Intervention:** A doctor tests if a new medication lowers blood pressure by prescribing it to one group of patients and a placebo to another. By comparing blood pressure levels, the doctor observes the medication's effects. Here, the doctor actively changes the treatment, altering the system.
- **Conditioning:** A doctor studies the blood pressure of patients already taking a specific medication by examining their medical records. The doctor does not change any treatments but focuses on patients who meet the condition of taking the medication. This is conditioning, where the system remains unchanged.

Counterfactual queries A *counterfactual query* for a given factual instance $\mathbf{x}^{\mathrm{f}}$ aims to estimate what would have happened had x_i instead taken value α. This effect is captured by the *counterfactual distribution* $p(\mathbf{x}^{\mathrm{cf}} \mid \mathbf{x}^{\mathrm{f}}, do(\mathrm{x}_i = \alpha))$, which can be computed using the *abduction-action-prediction* procedure by Pearl [111].

Counterfactual Fairness

An SCM also allows us to study a notion of fairness called *counterfactual fairness*, which tells us whether a decision is fair to an individual if the outcome in reality is the same as it would be in a "counterfactual" world in which the individual belongs to a different demographic, e.g., *would I have been selected if I were female?*.

Definition 5.1 (Counterfactual unfairness [77]) Let us consider $S \in [d]$ to be a sensitive attribute (e.g., gender) such that $\mathrm{x}_S \in \{0, 1\}$. Then, we can measure counterfactual unfairness (UF) of a binary classifier $\kappa : \mathbb{X} \to \{0, 1\}$ as

$$\text{UF} = \mathbb{E}_{\mathbf{x}^{\text{f}}} \left[P(\kappa(\mathbf{x}^{\text{cf}}) = 1 \mid do(\mathrm{x}_S = 1), \mathbf{x}^{\text{f}}) - P(\kappa(\mathbf{x}^{\text{cf}}) = 1 \mid do(\mathrm{x}_S = 0), \mathbf{x}^{\text{f}}) \right] \tag{5.2}$$

where $\mathbf{x}^{\text{cf}}$ is a counterfactual sample coming from the distribution $P(\mathbf{x}^{\text{cf}} \mid \mathbf{x}^{\text{f}}, do(\mathrm{x}_S = s))$, for $s \in \{0, 1\}$.

Then, the classifier κ is counterfactually fair [77] if $\text{UF} = 0$.

5.1.2 Variational Graph Autoencoder

Variational Autoencoders (VAEs) [67] are powerful latent variable models based on neural networks (NNs) for jointly i) learning expressive density estimators $p(\mathbf{x}) \approx \int p_\theta(\mathbf{x} \mid \mathbf{z}) p(\mathbf{z}) d\mathbf{z}$, where the likelihood function (a.k.a. *decoder*) is parameterized using a NN with parameters θ, and ii) performing approximate posterior inference over the latent variables $\mathbf{z}$ is possible via a variational distribution (a.k.a. *encoder*) $q_\phi(\mathbf{z} \mid \mathbf{x})$ parameterized using a NN with parameters ϕ. The parameters θ and ϕ can be learned by maximizing a lower bound on the log-evidence [19, 102, 115, 142].

Kipf and Welling [69] extend VAEs to account for prior graph structure information on the data [166], introducing the Variational Graph Autoencoders (VGAEs). VGAEs define a (potentially multidimensional) latent variable z_i per observed variable x_i, i.e., $\mathbf{z} := \{\mathrm{z}_1, \ldots, \mathrm{z}_d\}$. Additionally, VGAEs rely on an adjacency matrix $\boldsymbol{A}$, which is used by two GNNs, one for the encoder and one for the decoder, to enforce structure on the posterior approximation $q_\phi(\mathbf{z} \mid \mathbf{x}, \boldsymbol{A})$ and the likelihood $p_\theta(\mathbf{x} \mid \mathbf{z}, \boldsymbol{A})$. Hence, $\boldsymbol{A}$ determines which variables x_i influence z_j $\forall i, j \in [d]$ and vice versa.

5.2 Extended Abstract

Deep generative models are receiving increasing attention for causal queries in complex data [98, 106]. Existing approaches for causal inference focus on i) estimating the Average Treatment Effect (ATE)–a specific type of group-level causal queries–by assuming a fixed causal graph that includes a treatment variable [65, 94, 116, 131, 145, 173]; ii) discovering and intervening on the causal latent structure of the (e.g., image) data [65, 106, 107, 133, 159]; or iii) addressing interventional and/or counterfactual queries by fitting a conditional model for each observed variable given its

causal parents [45, 63, 75, 108, 109]. Within the scope of causality, GNNs have predominantly been used for causal discovery [166, 174] and only very recently, concurrently with us, exploited to answer interventional queries [172].

In Sánchez-Martín et al. [125], we investigate to which extent the inductive bias of GNNs—encoding the causal graph information—can be exploited to answer interventional and counterfactual queries. More specifically, to approximate the interventional and counterfactual distributions induced by interventions on a casual model. We assume i) causal sufficiency–i.e., absence of hidden confounders; and, ii) access to observational data and the true causal graph. We stress that the causal graph can often be inferred from expert knowledge [176] or via one of the approaches for causal discovery [48, 144]. We describe the architectural design conditions that a variational graph autoencoder (VGAE) must fulfill so that it can approximate causal interventions (*do-operator*) and *abduction-action-prediction* steps [111].

The resulting variational causal graph autoencoder, referred to as VACA, enables *approximating* the observational, interventional and counterfactual distributions induced by a causal model with unknown structural equations. Furthermore, VACA optimizes the observational distribution for all observed variables simultaneously to mitigate error propagation along the Markov factorization. We remark that parametric assumptions on the structural causal equations are in general not testable, thus may not hold in practice [113], and may lead to inaccurate results, if misspecified. VACA addresses this limitation by including uncertainty, i.e., a probabilistic model, in the estimation of the causal-parent relationships. Also, we carefully design the GNN networks of VACA, which parametrize the likelihood and approximate posterior distributions, to allow handling several aspects that often appear in real-world applications [35, 36]: partial causal knowledge and heterogeneous endogenous variables.

Partial causal knowledge. In some application domains the relationships between a subset of k_i endogenous variables $\mathbf{x}$ may be unknown, or they may be affected by hidden confounders. In such cases, we assume that set of k_i variables to be correlated and model them as one multidimensional and potentially heterogeneous node $\mathbf{x}_i = \{\mathrm{x}_{i1}, \dots, \mathrm{x}_{ik_i}\}$ that share the same latent random variable $\mathbf{z}_i$. This allows us to deal with a large variety of graphs in practice.

Heterogeneous endogenous variables. Heterogeneous causal nodes require us to model different functions for each node, i.e. nodes may now contain a mix of continuous/discrete variables. In general, GNNs are parametrized such that the parameters of the message function $\mathbf{f}_{\theta_m}$ and update function $\mathbf{f}_{\theta_u}$ are shared for

Table 5.1: Counterfactual unfairness (*uf*) and f1-score (*f1*) of a Support Vector Machine over 10 VACA seeds. Values multiplied by 100.

Metric	full	unaware	fair-x	fair-z
↑ *f1*	71.67	69.49	59.50	70.79 ± 5.15
↓ *uf*	14.01 ± 2.26	13.27 ± 2.28	0.14 ± 0.02	0.51 ± 0.19

all the nodes and edges in the graph. However, similar to the structural equations, we can define a unique set of parameters θ_{mij} for each message, so that we can model a different function for every edge in the causal graph. Further, we can also assume different update functions $\mathbf{f}_{\theta_{ui}}$ for each node i, by introducing different update parameters θ_{ui}. As a result, VACA fulfills the conditions to be a Neural Causal Model (NCM) Type 2 (Coll. 1 in Zečević et al. [172]) and thereof can represent the observational, interventional, and counterfactual distributions (Thm.1 and Thm. 3 in Xia et al. [155]).

We show in extensive synthetic experiments that VACA outperforms competing methods [63, 64] on complex datasets (in terms of the number of nodes and complexity of the structural equations). Unlike previous studies that focused solely on estimating the mean of the interventional/counterfactual distribution, VACA accurately captures the overall distribution, measured in terms of Maximum Mean Discrepancy [51]. Also, we analyze interventions on both root and non-root nodes, which was not always the case previously. Our code is publicly available at GitHub `https://github.com/psanch21/VACA`. Finally, we show a practical use-case in which VACA is used to assess counterfactual fairness of different classifiers trained on the real-world German Credit dataset [36], as well as to learn counterfactually fair classifiers without compromising performance.

Table 5.1 summarizes the results for the Support Vector Machine (SVM) classifier. The *full* column presents outcomes when all features of the German Credit dataset are used. Notably, the classifier achieves the best f1-score of 71.65 (top row); however, the counterfactual unfairness is also the highest at 14.01 ± 2.26 (bottom row). The *unaware* and *fair-x* columns reveal that excluding sensitive features enhances fairness but at the expense of performance. The *unaware* and *fair-x* columns show that by removing sensitive features we can improve the fairness at the cost of loosing performance. The right-most column, *fair-z*, uses the latent space of VACA, achieving competitive performance (70.79 ± 5.15) while maintaining counterfactual fairness close to zero (0.51 ± 0.19).

5.3 Discussion

Traditional statistical approaches uncover correlations (undirected relationships), whereas causal inference aims to identify causal relationships (directed relationships). The latter closely aligns with human reasoning. For instance, instead of merely observing a correlation between a treatment (T) and the curing of a disease (D), experts aim to assess whether a causal link exists between them. Causal inference [110], which involves answering interventional (e.g., "how would imposing stricter environmental regulations affect air quality in urban areas?") and counterfactual (e.g., "would this patient have different results if he received a different medication?" [160]) queries, is becoming more and more important. In many cases, the functional relationships between variables (e.g., between the amount of environmental regulation and air quality) are complex and unknown, making neural networks an attractive choice for modeling them.

Our work Sánchez-Martín et al. [125] complements concurrent research that theoretically studies the use of neural networks [155] and, more recently, GNNs [172], for causal inference. Our main contribution is the introduction of VACA, a variational causal autoencoder based on GNNs. We meticulously design VACA to capture the properties of SCMs by explicitly accounting for the dependencies between exogenous and endogenous variables in the causal graph induced by the SCM. Despite establishing a new state-of-the-art model for causal inference, VACA has limitations. Its ability to model complex structural equations, such as those in biology [123], is constrained by the architecture of the GNN encoder and decoder. For instance, aggregation functions may limit expressiveness [24]. Additionally, accommodating long causal paths would require increasing the depth of the decoder, potentially compromising GNN performance due to challenges associated with depth [42, 52, 89]. We expect VACA to benefit from advances in the field.

Drawing from insights gained from VACA, I co-authored a follow-up paper introducing CausalFlows [60]. This can be regarded as the normalizing flow counterpart of VACA. Here, we demonstrate that (partial) knowledge of the causal graph is crucial for designing autoregressive normalizing flows [66, 104, 105] that are *causally consistency* with the SCM, thereby enabling them to answer causal queries. CausalFlows outperforms existing baselines, requires less hyperparameter tuning, and is faster, thus establishing a new benchmark.

Future work

Future research directions for VACA include assessing its sensitivity to i) errors in the assumed causal graphs and ii) the presence of hidden confounders i.e., breaking the *causal sufficiency* assumption. Extending VACA to handle more complex causal models, such as non-DAGs, presents an exiting path for exploration. Also, it would be interesting to perform ablation studies on available GNNs architectures [154] for the encoder and decoder of VACA; as well as investigating how the performance deteriorates as we increase the length of the causal path and thus the required number of GNN layers [89]. Finally, it could also be promising to apply VACA to other causal questions such as privacy-preserving causal inference [78] or explainable machine learning [63].

Improving the interpretability of GNN predictions through conformal-based graph sparsification

6

This chapter is based on the publication Sanchez-Martin et al. [124]. Our objective is to advance the field of inherent interpretable and faithful graph classification, addressing the research question outlined in Q 3. In particular, we stick to the notion of interpretability understood as sparsity. In other words, we aim to use the minimal input information required to successfully tackle the task at hand.

6.1 Preliminaries

In this section, we provide an introduction to the main two concepts that we rely on to build the proposed model: (graph) reinforcement learning and conformal predictions.

6.1.1 Reinforcement learning

Reinforcement Learning (RL) is a learning paradigm in which an agent learns to optimize decisions by interacting with an environment [138]. Figure 6.1 illustrates this process. An RL problem is typically modeled as a Markov Decision Process (MDP), defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, $\mathcal{P}$ the transition probability, R the reward function, and γ the discount factor. We are interested in a subfield of RL algorithms whose objective is to find a policy π_ϕ that maximizes the expected cumulative reward.

Policy Gradient Methods Policy gradient methods directly optimize the policy using gradient ascent on the expected cumulative reward [139]. Proximal Policy Optimization (PPO) [130] is a policy gradient method that introduces an objective function fostering exploration while mitigating drastic policy updates. PPO aims to solve the optimization problem:

$$L^{CLIP}(\phi) = \mathbb{E}_t \left[\min \left(r_t(\phi)\hat{A}_t, clip(r_t(\phi), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] \quad (6.1)$$

where ϕ are the learnable parameters of the policy $\pi_\phi(a_t|s_t)$, $r_t(\phi) = \frac{\pi_\phi(a_t|s_t)}{\pi_{\phi_{old}}(a_t|s_t)}$ is the ratio between the probability of the action at time t using the old and the updated parameters of the policy

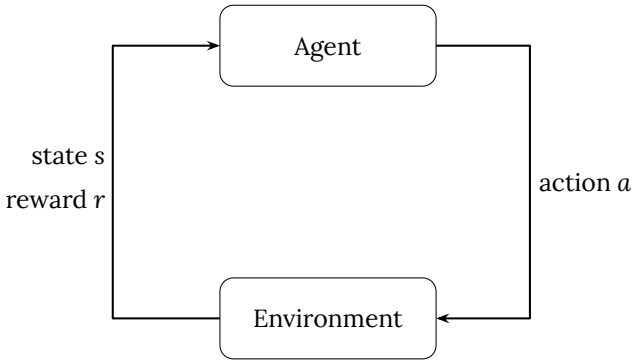


Figure 6.1: RL diagram. A schematic representation of the reinforcement learning loop. The *Agent*, typically instantiated as a policy π_ϕ, receives a state s and reward r, subsequently generating the action a. The *Environment* processes this action and provies the agent with a new state and reward.

π, $\hat{A}_t$ an estimator of the advantage function * at time t, and $\epsilon \geq 0$ a hyperparameter controlling the deviation from the old policy.

RL in graphs Graph Reinforcement Learning (GRL) integrates graph learning with RL consisting on graph-structured environments. This combination introduces additional challenges in the design of the state space, action space, reward funcion and policy architecture [101]. In this setting, the state space usually refers to a graph and the action space can refer to the nodes, edges, or the entire graph. GNN have emerged as promising tools for parameterizing policies as they are general enough to handle any of these scenarios.

6.1.2 Conformal Predictions

Conformal prediction [6] is a framework that rigorously quantifies uncertainty in machine learning predictions. Given a labeled dataset $\{\mathbf{x}_i, y_i\}_{i=1}^N$, a heuristic measure of uncertainty of our predictor (e.g., softmax values from a classifier $\mathbf{f}_\theta$), and a scoring function $s(\mathbf{x}, y) \in \mathbb{R}$ that reflects prediction uncertainty, conformal prediction generates a prediction set for a new input x_{test} as follows:

$$\mathcal{C}\left(x_{\text{test}}\right) = \left\{y : s\left(\mathbf{x}_{\text{test}}, y\right) \leq \hat{q}\right\} \subseteq [K]. \tag{6.2}$$

Here, $\hat{q}$ represents the $\frac{\lceil (n+1)(1-\alpha) \rceil}{n}$ quantile of the calibration scores $\{s_i = s(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $\alpha \in [0, 1]$ is a predefined error rate. In the context of classification tasks, one commonly used conformal

* The advantage function essentially measures how much better or worse it is to take action a in state s compared to the average expected return of all actions in state s under the policy π.

procedure is Adaptive Prediction Sets (APS) [7, 118], which defines the scoring function as:

$$s(\mathbf{x}, y) = \sum_{j=1}^{k} \mathbf{f}_\theta(\mathbf{x})_{\pi_j(\mathbf{x})}, \text{ where } y = \pi_k(\mathbf{x}) \tag{6.3}$$

Here, $\pi(\mathbf{x})$ represents the permutation of $[K]$ that arranges the softmax values $\mathbf{f}_\theta(\mathbf{x})$ from most likely to least likely.

6.2 Extended Abstract

GNN is a powerful family of deep learning models that has achieved state-of-the-art performance in solving graph classification tasks where the goal is learn a function $\mathbf{f}_\theta : \mathcal{G} \mapsto \hat{y}$. However, the fact that they are black box-models and that most GNN architectures aggregate information from all nodes and edges in a graph—regardless of their relevance to the task at hand—leads to one of their principal shortcoming: a lack of human-interpretable predictions. This opacity hinders their potential for practical, real-world impact, as practitioners often require interpretable models to inform decision-making, ensure trustworthiness, and comply with regulations [33]. We argue that interpretability in this context can be achieved with graph sparsity. That is, using a subgraph $\mathcal{G}_s \subset \mathcal{G}$ with a minimal set of nodes and edges from the graph $\mathcal{G}$ for prediction.

This viewpoint of interpretability is exemplified in Figure 6.2 with the synthetic *BA2Shapes* dataset [163]. In this dataset, a binary classification task can be solved using only specific motifs of the graphs (the house or the cycle motifs, i.e., yellow nodes), while the original graphs contain additional irrelevant information for the task at hand, i.e., purple nodes. A sparser graph reduces the volume of information used for making predictions, rendering it easier for humans to understand the GNN predictions. It is crucial to note that interpretability is achieved through sparsity only when the subgraph completely excludes information from the omitted nodes and edges since only then the subgraph, and thus, the explanation is *faithful* to the model prediction. This situation does not arise, for example, if message passing [68] is applied to obtain the node embeddings before finding the subgraph.

While most of explainability methods for GNNs are post-hoc [95, 103, 128, 163, 167, 168, 170], some existing approaches attempt to identify the predictive subgraph during training [20, 44, 74, 83, 136, 162]. These methods, however, exhibit two primary limitations: i) they often still rely on the entire graph for predictions [74, 83,

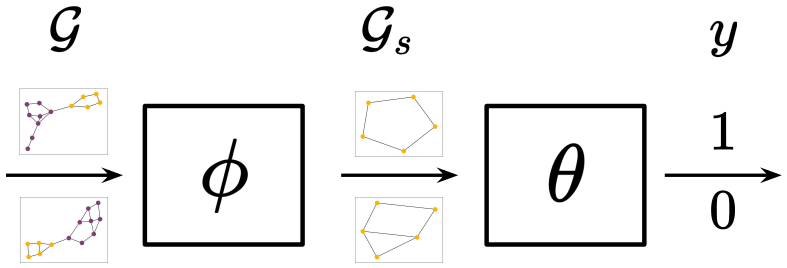


Figure 6.2: CORES pipeline. Illustration of the pipeline of CORES using the synthetic *BA2Shapes*. On the left, we present two examples of original graphs ($\mathcal{G}$) corresponding to the positive class (top, cycle motif) and negative class (bottom, house motif). The policy ϕ of CORES takes $\mathcal{G}$ and sparsifies it, resulting in the predictive subgraph $\mathcal{G}_s$, which retains only the relevant information for the task, i.e., the motifs. Finally, the graph classifier θ takes $\mathcal{G}_s$ as input and produces the prediction $y \in \{0, 1\}$.

162], resulting in a lack of *faithfulness* to the model prediction (as described above), and/or ii) they impose strong assumptions concerning the structure of the predictive subgraph, such as a pre-defined size [20, 44, 74, 136]. In practical scenarios, relaxing assumptions on the subgraph structure is desirable. For instance, Debnath et al. [27] observed that diverse "motifs" of chemical elements are predictive of mutagenic effects in compounds.

In Sanchez-Martin et al. [124] we propose CORES a GNN *training* approach that jointly i) finds the most predictive subgraph $\mathcal{G}_s \subset \mathcal{G}$ by removing edges and/or nodes—*without making assumptions about the subgraph structure*—while ii) optimizing the performance of the graph classification task $\mathbf{f}_\theta$ solely using the selected subgraph $\mathcal{G}_s$, thereby enhancing the interpretability of the classifier.

More concretely, given a labeled dataset $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_i$, the main goal of CORES is to identify a compact predictive subgraph $\mathcal{G}_s \subseteq \mathcal{G}$ that maintains high performance for a graph classification task. To this end, we use a bi-level optimization approach.

$$\phi^* = \arg\min_{\phi} \mathcal{L}_{\text{spa}} \left(\theta^\star(\phi), \phi, \mathcal{D}^{val} \right) \tag{6.4}$$

$$\text{s.t. } \theta^\star(\phi) = \arg\min_{\theta} \mathcal{L}_{\text{perf}} \left(\theta, \phi, \mathcal{D}^{tr} \right). \tag{6.5}$$

The performance optimization refers to Equation (6.5) which represents a standard graph-supervised problem. Here, the objective is to minimize a loss function $\mathcal{L}_{\text{perf}}$, e.g., the cross-entropy loss. Thus the goal is to learn a function $\mathbf{f}_\theta : \mathcal{G}_s \rightarrow \hat{\text{y}}$ with parameters θ that minimizes the prediction loss. The sparsity optimization corresponds Equation (6.4). Here, we to learn a function $\pi_\phi : \mathcal{G} \mapsto \mathcal{G}_s$ responsible for identifying the predictive subgraph. This function corresponds to the policy of the reinforcement learning component of CORES. It presents a considerable challenge due to the combinatorial nature of the node and edge removal process. Specifically,

we must determine which nodes ($v \in \mathcal{V}$) and/or edges ($(u, v) \in \mathcal{E}$) should be removed. Drawing inspiration from SparRL [150], we formulate the sparsification task as a Markov Decision Process (MDP) and address it through the framework of graph reinforcement learning [101]: we parameterize the policy π_ϕ using a GNN. GNN formulation is general enough to account for edge and node removal. In contrast to prior approaches that rely on value-based methods like Deep Q-Learning [97], we opt for the policy gradient method Proximal Policy Optimization (PPO) [129] to capture the inherent uncertainty in the sparsification process.

The reward function R plays a pivotal role in shaping the policy's behavior. It should provide positive rewards for predictive and sparse subgraphs while penalizing those that negatively impact the performance of the graph classifier. It is also important to consider that the classifier makes errors, i.e., it does not achieve perfect performance. Consequently, there are instances where a subgraph may be genuinely predictive, but the classifier produces an incorrect prediction. Our reward design accounts for this inherent uncertainty through the use of conformal predictions [6]. We define it as follows:

$$R = \begin{cases} \lambda R_p + (1-\lambda) R_s(\mathcal{G}_s) & \text{if } y \in \mathcal{C}(\mathcal{G}_s) \wedge |\mathcal{C}(\mathcal{G}_s)| = 1, \\ \frac{R_p}{|\mathcal{C}(\mathcal{G}_s)|} & \text{if } y \in \mathcal{C}(\mathcal{G}_s) \wedge |\mathcal{C}(\mathcal{G}_s)| > 1, \\ -R_s(\mathcal{G}_s) & \text{if } y \notin \mathcal{C}(\mathcal{G}_s). \end{cases} \tag{6.6}$$

Here, R_p refers to the performance component, R_s is the sparsity promoting component and $\mathcal{C}(\mathcal{G}_s)$ is the prediction set when using $\mathcal{G}_s$ as input to the classifier. Our carefully designed reward function allows us to introduce inductive biases toward either sparse or high-performing solutions

We conduct an ablation study to show the impact of different values of λ and maximum desired ratio (d) on performance, as well as the effect of the choice of base GNN architecture. We also perform an empirical comparison on nine different graph classification datasets to show that our method is competitive in performance with the baselines while relying on significantly sparser subgraphs, leading to more interpretable GNN-based predictions.

6.3 Discussion

Interpretability is becoming a critical aspect of deep learning, particularly for models deployed in real-world decision-making scenarios, such as drug re-purposing [146]. Graph Neural Networks

(GNNs) have demonstrated remarkable performance in graph-level tasks, such as estimating chemical properties of molecules [151]. However, their complexity often leads to interpretability challenges and thus, a lack of deployment. In this context, the removal of irrelevant information from the input—i.e. *sparsifying* the input—improves interpretability. The underlying rationale behind this approach is intuitive: reducing the input size reduces the cognitive load on experts.

The main contribution of this thesis in this area is the introduction of CORES in Sanchez-Martin et al. [124]. CORES is a novel GNN training that provides *faithful* graph classification prediction using minimal predictive subgraphs. CORES achieves this goal by simultaneously finding the irrelevant edges and/or nodes to be removed, without imposing assumptions about subgraph structures; and optimizing the performance of the graph classification task. The key novelty behind CORES is the design of a reward function that i) takes into account the uncertainty of the classifier in the reward assignment and ii) allows practitioners to introduce inductive biases towards either sparse or high-performing predictive subgraphs. Our empirical evaluation, conducted on nine graph classification datasets, provides evidence that our approach not only matches the performance of baselines using complete graph information but also relies on significantly sparser subgraphs. Consequently, GNN-based predictions obtained with CORES are more interpretable, addressing the primary motivation behind our work.

Future work

Future research could focus on improving the efficiency of the reinforcement learning module to accelerate the training and inference process of CORES. Presently, training CORES involves collecting batches of subgraphs for each original graph—i.e., collect episodes—and performing gradient updates on the policy parameters (i.e., the network responsible for identifying optimal subgraphs) and graph classifier parameters. This involves two independent backpropagation steps and results in a notable time overhead of one or two orders of magnitudes compared to baselines not using reinforcement learning. Similarly, during inference, CORES requires a forward pass on the policy to obtain the subgraph followed by a forward pass on the graph classifier to get the final prediction. Here, thus, CORES also introduces a smaller yet significant time overhead. Exploring methodologies to speed up CORES presents an interesting avenue for future research, such as improving the data efficiency of the reinforcement learning component or parallelizing the computation of the graph sparsification

for mini-batch selection.

Furthermore, alternative approaches to using reinforcement learning for subgraph finding, such as meta-heuristics [39], could be explored. Although these approaches do not require gradient updates (no training step needed), they typically exhibit slower inference times. An additional interesting direction for future work involves refining the current reward function to penalize specific types of errors or undesirable graph structures, such as isolated nodes.

Moreover, exploring the application of CORES to graph regression tasks could be promising. The main change to the current setup would be slight modifications in the reward function. Particularly, finding a suitable conformal prediction method [6] and performance component, e.g., measure proximity to the true value.

Finally, we hypothesize that the obtained subgraphs may mitigate the oversquashing phenomenon, thus facilitating the more effective capture of task-relevant information coming from interactions among distant nodes. Nevertheless, this should be formally studied and presents an interesting research direction.

Part III

Conclusion & Open Questions

Conclusion 7

In this concluding chapter, we present a comprehensive overview of the primary findings, insights, and contributions to the field of GNNs provided by the three papers included in this thesis. Then, we provide a summary of the key research areas and outline interesting open questions that are promising for future research. Lastly, we offer concluding remarks.

7.1 Overview of Findings & Impact

Graph Neural Networks have certainly revolutionized the field of graph learning. Over the last decade, dozens of GNN architectures have emerged [154, 177]. Unfortunately, as presented in Chapter 4 of this dissertation, the effectiveness of these architectures depends on the nature of the data and the task at hand, making it challenging—if not infeasible—to determine a priori the optimal choice. The search for optimal architectures requires extensive cross-validation, which consumes significant hardware resources, energy, and researcher time that could be spent on higher-value tasks such as literature review or idea discussion. Additionally, inherent issues such as oversmoothing [121] and oversquashing [31], coupled with practical barriers such as data scarcity and the stochasticity in gradient-based learning, further complicate identifying optimal configurations.

Throughout this thesis, we have shown the importance of introducing *explicit inductive biases* in GNN-based models and optimization processes in order to overcome these challenges. Such inductive biases should aim to narrow the search space and steer optimization towards solutions that achieve competitive performance in the target task. Our overall objective has been to introduce practical methodologies that advance GNN research in three pivotal domains for the real-world implementation of GNN models: efficient architecture search (Q 1), causal inference (Q 2), and inherent interpretable models (Q 3). Efficient architecture search is an important topic since we have limited resources. As machine learning gains broader adoption, there is an increasing need to reduce the energetic footprint of training ML models. Causality plays a pivotal role as it enables reasoning about the consequences of changes applied to the generative process of data (interventional reasoning) and modeling the consequences of changes retrospectively while accounting for what happened (counterfactual reasoning). These

are tasks that statistical pattern recognition learning cannot tackle. Finally, the significance of inherently interpretable models is that they enable domain experts to grasp the process behind predictions, while also helping to uncover systematic errors and biases during training, i.e., before the models are deployed. Together, these aspects should contribute to the widespread adoption of GNNs for tasks where they have already achieved state-of-the-art results or are a promising methodology.

Q 1 How can we reduce the need for GNN architecture cross-validation?

One of the fundamental challenges in GNN research is the large number of architectures and hyperparameters that need to be cross-validated. This makes the search for optimal configurations a tedious process, consuming significant time for researchers, as well as computing resources.

With the introduction of L-CAT in Chapter 4, we show—for the first time in the GNN community—that *learning to interpolate* reduces the need for cross-validation of the GNN architectures. The proposed approach intervenes only in the adjacency matrix and is therefore general enough to be extended beyond the considered GNN architectures and to be used in a wide range of GNN-based applications.

Q 2 How to design GNN-based models for causal inference?

Finding and quantifying cause and effect is closely related to human reasoning and decision-making. GNNs provide a natural framework for modeling directed graphs, making them suitable for causal inference, i.e., answering interventional and counterfactual queries. However, their off-the-shelf applicability is limited because the message-passing procedure may introduce dependencies that are not present in a given causal graph.

In Chapter 5, we introduce VACA to address this limitation. We carefully impose constraints and requirements on the GNN architectures that parameterize the approximate posterior distribution and the likelihood distribution. This ensures that VACA captures the correct dependencies given by the causal graph and is able to approximate causal queries. This work represents the first comprehensive study of the *design requirements* of GNN architectures tailored for causal inference. In the evaluation, we focus on two important tasks of *responsible machine learning*: training counterfactually fair classifiers and assessing the counterfactual fairness of a classifiers. In my follow-up work Javaloy et al. [60] we go one

step further and provide some identifiability results. However, it is crucial to recognize that our work, like much of the causal inference literature, relies on strong assumptions about the underlying causal model, in particular the absence of hidden confounders (causal sufficiency assumption). Thus, we emphasize the importance for practitioners to rigorously validate model assumptions, especially when using their model in real-world applications.

Q 3 How to improve inherently interpretable GNN graph classification?

Another important practical aspect is the development of GNN models capable of providing *inherently interpretable predictions* for domain experts. That is predictions that are generated in a way that is consistent with the reasoning processes that individuals knowledgeable in the specific domain would follow. However, the black-box nature of GNNs often presents a significant challenge in achieving interpretability. Also, quantifying the interpretability of predictions is itself a complex problem that lacks sufficient constraints or information to determine a unique solution. Moreover, interpretability is often a subjective matter, which adds to the complexity of the problem [17].

In Chapter 6 we focus on a particular notion of interpretability that is easy to quantify: we aim for *sparsity*. We argue that the more information in the input to the model, the more challenging the understanding of the prediction becomes. In other words, reducing the information in the input makes the predictions easier to understand. In particular, we focus on graph classification tasks. In this setting, reducing the input means minimizing the number of edges and/or nodes used by the model as input. To achieve this objective, we propose a novel training procedure, which we call CORES, that achieves competitive performance while using a small input graph. The core of our approach lies in using reinforcement learning for selecting the nodes/edges to keep. The effectiveness of this selection process depends heavily on an informative reward assignment, e.g. assigning high rewards to input graphs that are both minimal and discriminative. To facilitate this, we carefully design a reward function that leverages conformal predictions to ensure that high rewards are assigned only when the classifier is certain that the prediction is correct. The impact of this work lies in showing the potential of combining concepts from different fields to address the challenge of finding a predictive subgraph, an NP-hard problem, in GNN-based graph classification tasks in order to improve the interpretability of predictions. More generally, I hope that this work will also contribute to increasing interdisciplinary collaboration. I believe that fostering conversations between different communities

has immense potential to overcome domain-specific challenges and drive innovation.

7.2 GNNs Today and Open Challenges

Research on GNNs has been consistently featured in major ML conferences such as NeurIPS, ICML, or ICLR, covering a wide range of topics. Some of these include exploring higher-order GNNs tailored for hypergraphs [37]; addressing oversmoothing and oversquashing phenomena, namely understanding [31, 153] or mitigating [8, 122] them; exploring temporal graph learning [58]; integrating equivariance principles into GNNs [34]; studying neural architecture search (NAS) for GNNs [157]; and incorporating interpretability into GNNs [92, 169].

These represent focal areas that I have consistently and extensively encountered throughout my recent years of research. Among these topics and others, the GNN community still faces challenges, some of which I faced during my PhD.

Reducing Hyperparameter Tuning

The challenge of reducing cross-validation not only posed Q 1 but also was a recurring bottleneck throughout the research conducted in this dissertation, hindering progress. The prevalent approach to finding the optimal configuration of hyperparameters is grid search. This process involves specifying the hyperparameter set and the corresponding values for cross-validation—such as $[1e^{-3}, 1e^{-2}]$ for the learning rate and $[16, 32]$ for the dimension of the hidden space—before exploring all possible combinations. Essentially, if we have h different hyperparameters and want to explore v values for each, this results in v^h unique combinations. It is clear that both h and v must be $<< 10$ to effectively manage computational resources.

Unfortunately, it is often not possible to meet these constraints. Table 7.1 shows some of the most common hyperparameters and their corresponding values that we, and the broader GNN community, typically need to cross-validate. Assuming a"good case scenario" where only two values per hyperparameter need to be cross-validated, there would be 2^{16} potential combinations. It becomes evident that cross-validating all these combinations, let alone running different parameter initializations (a.k.a. runs or seeds) for each, is not feasible. Note that training a single GNN model can take anywhere from minutes to hours. While this dissertation focuses on GNNs, this challenge is ubiquitous in deep

Hyperparameter name	Set of values
GNN architecture	{GCN, GAT, GIN, PNA}
Parameter of the GNN architecture	E.g., number of heads (GAT) or ϵ (GIN)
Number of hidden layers	$\{0, 1, 2, 3, 4, 5\}$
Dimension of the hidden space	$\{8, 16, 32, 64, 128, 256\}$
Skip-connection	{True, False}
Global pooling type	{mean, std, max, min, attention}
Dropout rate	$\{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$
Layer normalization	{True, False}
Batch normalization	{True, False}
Early stopping patience	$\{1, 5, 10\}$
Early stopping delta	$\{0.0, 0.001\}$
Batch size	$\{16, 32, 64, 128\}$
Optimizer type	$\{Adam, SGD\}$
Initial learning rate	$\{1e^{-4}, 1e^{-3}, 1e^{-2}, 1e^{-1}\}$
Scheduler type	{ReduceLROnPlateau, ExponentialLR}
Parameter of the scheduler	E.g., step size

Table 7.1: Typical hyperparameters cross-validated for GNN model This table presents some (but not all) of the hyperparameters that are usually cross-validated in experimental analysis for GNN models.

learning. For instance, the PPO [130] has over 5 hyperparameters that significantly impact performance and require cross-validation [101].

In practice, practitioners must ultimately decide which hyperparameters to fix and which ones to cross-validate. Usually, this process is somewhat subjective, relying on values deemed effective by previous research without certainty regarding their optimality for the given task. Frankly speaking: this procedure lacks scientific rigor, consumes substantial economic resources (hardware and energy), and demands significant researcher time. Therefore, I believe that it is important for the GNN community to invest more effort in developing and adopting methodologies to reduce this burden. Yang and Shami [158] showed that using Hyperparameter Optimization (HPO) techniques can lead to similar or better performance in multiple machine learning tasks—such as classification with Random Forest or Support Vector Machines— with up to ten times less computing resources. HPO is standard in fields where many hyperparameters significantly affect performance, such as Reinforcement Learning [38]. Because of this, I believe that the use of HPO algorithms holds significant promise in mitigating computational burdens and advancing scientific progress in the field of GNNs. Methods range from simple techniques like random search [14] to more sophisticated Bayesian [135] or bandit-based approaches [87]. Several publicly available tools, such as WandB

Sweeps * or Optuna [3], can simplify the integration of HPO into projects. Thus, I encourage the GNN community to explore these methodologies.

Key challenges in interpretable machine learning

The interest in interpretability within GNNs has surged in recent years, leading to a significant increase in research papers. Major AI conferences frequently host workshops dedicated to this topic. However, the field faces several fundamental challenges that need to be addressed.

Clearly define what interpretability question is being addressed Methods to address interpretability often involve mathematically complex procedures, frequently requiring the definition of auxiliary models and training procedures [57, 161], and/or analysis of gradients [132]. Additionally, interpretability covers a multitude of facets open to study. As highlighted by Bordt and Luxburg [17], interpretability questions can be broadly categorized into non-technical, such as *Is the model producing this output because it has seen the Harry Potter novels during training?*, and technical, such as *Is the explanation algorithm robust to perturbations?* Although both categories of interpretability questions are significant and valid, rigorous analysis is typically only feasible for technical ones. In addition, the lack of clarity regarding the specific interpretability questions being addressed, combined with the complexity of methodologies, can lead to misinterpretation of the generated explanations (i.e., outputs). This hinders progress tracking and slows down advancements in the field. Hence, I argue that the precise definition of the interpretability question at hand is an area where the community should invest greater effort.

Inherent interpretability Inherent interpretability, also known as in-training interpretability, plays an important role in the deployment of GNN models. By integrating interpretability directly into the model architecture or training process, several benefits can be achieved. First, it ensures that the predictions are faithful to the model. Second, it facilitates the identification and mitigation of biases and errors during model development, thereby enhancing fairness and accountability. Third, it fosters collaboration between domain experts and researchers by making it easier for domain experts to provide feedback and insights throughout the model development process. Fourth, it removes a step in the model deployment pipeline: the post-hoc interpretability step.

* https://docs.wandb.ai/guides/sweeps

Finally, inherent interpretability often results in ML models that can be more easily accepted and trusted by users and stakeholders [4]. I find sparsity to be a simple and promising methodology. However, I found several limitations when trying to achieve it during the development of CORES (see Chapter 6). On the one hand, I encountered disparate behavior in the graph sparsification component when slightly changing the definition of the reward function. For instance, not penalizing keeping the original graph sometimes resulted in the sparsification process removing nothing. Intuitively, in such cases, the reward function incentivized solutions that prioritized performance while neglecting sparsity. In the graph reinforcement learning (GRL) community, it is widely acknowledged that the choice of reward function significantly impacts algorithmic success. It is also acknowledged the challenge of crafting informative reward functions [101]. This task typically requires significant domain expertise, iterative experimentation, and usually lacks any theoretical guarantees. *How can we automate the design process of rewards? How do we ensure their robustness to corner cases? Which additional information beyond downstream predictor performance can we leverage?* These questions remains unanswered and represent interesting areas of research within the GRL community. On the other hand, subgraphs may not be interpretable by a human. For example, in binary graph classification, one might solve the task by keeping two nodes for the positive class and one node for the negative class after graph sparsification. However, this simplistic approach may lead to a classifier that simply learns to count nodes, overlooking the valuable information contained within the nodes themselves. Such an approach often provides the wrong reasoning behind the predictions and leads to errors in interpretation. Imposing constraints on the expected subgraph cannot be easily automated, and presents an interesting area for further research.

Evaluation of post-hoc interpretability The evaluation of post-hoc interpretability methods presents significant challenges as it relies on the quality of both the dataset and underlying GNN predictor. On the one hand, we have dataset-related failure cases. These can arise due to the presence of spurious correlations that do not represent the underlying reasoning required for task resolution. For instance, in a graph classification scenario, correlations between graph density and labels may exist, where graphs from the positive class exhibit higher density than those from the negative class. Despite these correlations not constituting true reasoning for task resolution, GNN models may wrongly learn to rely on them, entangling the evaluation of post-hoc interpretability methods. On the other hand, we have failure cases related to the GNN predictor itself. For example, the model may lack the necessary

expressive power to effectively solve the task, such as insufficient layers, or may only be capable of capturing a single rationale, e.g., only be able to capture a certain graph structure. Additionally, failures in optimization due to local stochastic optimization, further entangle and complicate the evaluation of interpretability methods. Determining whether interpretability methods effectively work and yield conclusive results becomes inherently difficult if we cannot assure that the underlying GNN relies on the correct rationale. To address these challenges, it is crucial to isolate the success or failure of post-hoc methods from the other parts of the machine learning pipeline: the dataset and the black-box model. One promising strategy is the development of synthetic datasets and GNN predictors, which provide researchers complete control over the involved components, enabling a more robust evaluation of interpretability approaches. Further research in this direction holds promise in enhancing the evaluation of post-hoc interpretability methods, thereby advancing the field [1, 96].

Complexity overhead In Chapter 7, we showed that CORES introduces a notable time overhead when compared to baseline models, affecting both training and inference processes. This challenge is ubiquitous across inherent interpretable GNN models [92] and highlights the trade-off between interpretability, performance, and computational efficiency. Addressing this trade-off remains an open question in the field. In particular, how to enhance all three aspects simultaneously. Advances in this regard are important for the widespread deployment of interpretable models in real-world applications. It is essential to note that interpretability often does not directly translate into increased value for stakeholders—e.g., a better content recommendation for end-users or increased profit for investors. Therefore, it is important that interpretability methods do not compromise performance or increase deployment costs.

Challenges in causal inference

Beyond causal sufficiency One of the prevailing assumptions in causal inference is causal sufficiency, which means that external causes (exogenous variables) are independent of each other. However, this assumption often does not hold in real-world scenarios. For instance, when studying the causal relationship between education level and income, it is commonly accepted that higher education leads to higher income. Nonetheless, hidden variables can confound this direct cause-effect relationship. Factors such as innate ability, family background, and access to resources may influence both educational achievements and income levels. Failing to account for these hidden variables can lead to wrong estimates of

the direct causal effect of education on income. Relaxing the causal sufficiency assumption and accounting for hidden confounders constitutes a promising research direction.

Causal Inference with Temporal Data Temporal data inherently provides some notion of causality. Let us define a time series as $\mathbf{t}_i = \{t_{i1}, \ldots, t_{iT}\}$ with T elements sorted by timestamp. Then, given a set of time series $\{\mathbf{t}_1, \ldots, \mathbf{t}_d\}$, we can compute the *Granger causality* among them [134]. In other words, we can determine whether a time series $\mathbf{t}_i$ helps in forecasting another time series $\mathbf{t}_j$. In this context, leveraging dynamic GNNs [62] holds considerable potential to address causal queries. Analogous to the static counterpart setting introduced in Chapter 5, the effective design of GNN architectures becomes crucial to ensure the dependencies within the assumed (or discovered) causal graph are maintained. This presents an interesting avenue for further research.

Uncertainty quantification

Performance and certainty are two essential aspects of any predictor, ideally exhibiting both properties simultaneously. However, this is often challenging. High performance can be achieved by embracing uncertainty, for instance, by providing all possible outcomes as the prediction, albeit at the cost of usefulness, a.k.a. *efficiency*. Finding a predictor that simultaneously achieves high performance and certainty is nontrivial. In domains with high stakes, such as autonomous driving, healthcare, and legal contexts, uncertainty quantification holds paramount importance since predictions carry significant consequences for human life. For instance, in legal contexts, predictive uncertainty can help determine guilt or innocence in criminal cases. Similarly, in drug development, accounting for uncertainty regarding drug effectiveness and potential side effects can lead to improved patient safety during clinical trials and regulatory approval processes. What is more, regardless of its relevance to the task at hand, quantifying uncertainty can provide insights into the inner workings of the model. For instance, it can reveal that certain areas in the input domain (e.g., certain demographics) exhibit greater uncertainty, which exposes discrimination issues. In recent years, interest in uncertainty quantification has significantly increased [13, 16, 30, 56, 71, 86, 100, 137, 147, 148]. However, the focus on uncertainty quantification within the GNNs field has been relatively limited [56, 171]. Conformal predictions, as discussed in Chapter 6, offer one approach for uncertainty quantification, but further exploration in this area is needed. Some interesting questions remain unanswered: *How can we develop conformal prediction methods for non-exchangeable data (e.g.,*

for link prediction tasks)? How can we train GNN models to achieve desirable conformal prediction properties, e.g., efficiency or conditional coverage? These offer interesting research directions.

Unexplored use-cases

During my PhD I mainly focused on standarized tasks within the GNN community, such as graph or node classification. For evaluating the proposed models on these tasks, I used widely accepted datasets from Torch Geometric [40] and the Open Graph Benchmark [55]. Unfortunately, these datasets primarily cover a limited set of domains. Datasets for graph classification mainly come from the biomedical domain, while datasets for node classification are predominantly from online shopping or academic research domains. This narrow focus is surprising given the ubiquity of graph data in diverse domains such as:

- **Transportation Networks:** Nodes represent stations, stops, or intersections; edges represent routes or roads.
- **Telecommunications:** Nodes represent communication devices (e.g., phones, computers) or servers; edges represent communication links.
- **Cybersecurity:** Nodes represent computers, servers, or users; edges represent connections, data flows, or known attacks.
- **Supply Chain Networks:** Nodes represent suppliers, manufacturers, distributors, or retailers; edges represent the flow of goods or finances.
- **Healthcare:** Nodes represent patients, healthcare providers, or medical entities; edges represent treatment interactions, patient histories, or disease transmissions.

These domains (and others) are often overlooked when benchmarking new GNN architectures, primarily due to the lack of available datasets or the fact that available datasets are unknown within the community. The limited focus on specific domains presents challenges when attempting to explore new areas.

During my PhD, I studied *ambient awareness*, a concept unexplored by the GNN community and more broadly by the Machine Learning and Data Science communities. Ambient awareness (AA) refers to the development of awareness about *who knows who* and *who knows what* that users of online social networks experience simply by being exposed to a vast stream of pieces of digital information, e.g., tweets on Twitter (recently rebranded as X.com) [85, 140]. Interestingly, AA happens in the absence of extensive one-to-one communication and without the awareness being the ultimate goal. These properties make AA a compelling mechanism for information acquisition and an interesting subject of study.

My work Sanchez-Martin et al. [126] was the first data-driven study of AA. Previous research, based on user surveys [85], indicated that individuals report ambient awareness only for parts of their network. It was unclear whether this limitation was due to cognitive capacity or limited exposure to diagnostic content (e.g., tweets). My findings suggest that the latter is the primary constraint. This research represents an initial step in the data-driven study of AA, laying the groundwork for future investigations. However, it does not present any "learning" as it is purely a data analysis work.

In unpublished follow-up work, I unsuccessfully attempted to use GNN to predict the existence of ambient awareness between Twitter users. I faced significant data-related challenges: identifying relevant data, collecting and cleaning it, ensuring it contained sufficient predictive information (high signal-to-noise ratio), and selecting appropriate GNN architectures. These questions are typically already answered when developing new GNN architectures for well-established tasks and domains.

Conducting research in a new use-case is challenging, especially without available datasets and tested models. Establishing broader standardized benchmarks across diverse domains in GNN research can expand the impact of GNNs and uncover new interesting research questions and limitations.

7.3 Closure

In this dissertation, we advanced the field of GNNs by introducing practical approaches that tackle important challenges for the implementation of GNNs in real-world applications: *efficient architecture search*, *causal inference*, and *inherent interpretability*. We introduced to the community the potential of learnable interpolation to mitigate cross-validation demands associated with the diversity of GNN architectures, opening a new line of research in the *efficient search of GNN architectures*, We also presented GNN-based solutions for performing *causal inference* and generating *interpretable* and faithful predictions for graph classification—two subfields that are important for wider adoption of GNN models. Furthermore, we offer open-source implementations of all proposed methods and discuss the limitations of each of the approaches. We hope that the methodologies discussed in this thesis will inspire future research aimed at i) reducing computational resource requirements and/or giving importance to the explicit inductive biases in the development of GNN models.

Part IV

Appendix

Publications A

Here, we include the complete publications (main manuscript and appendices) that are part of this dissertation. In order, we include the publications corresponding to L-CAT [59], VACA [125], and CORES [124].

Learnable Graph Convolutional Attention Networks

Adrián Javaloy [1,*] **Pablo Sánchez-Martín** [1,2,*] **Amit Levi** [3] **Isabel Valera** [1,4]

[1]Department of Computer Science of Saarland University, Saarbrücken, Germany
[2]Max Planck Institute for Intelligent Systems, Tübingen, Germany
[3]Huawei Noah's Ark Lab, Montreal, Canada
[4]Max Planck Institute for Software Systems, Saarbrücken, Germany

Abstract

Existing Graph Neural Networks (GNNs) compute the message exchange between nodes by either aggregating uniformly (*convolving*) the features of all the neighboring nodes, or by applying a non-uniform score (*attending*) to the features. Recent works have shown the strengths and weaknesses of the resulting GNN architectures, respectively, GCNs and GATs. In this work, we aim at exploiting the strengths of both approaches to their full extent. To this end, we first introduce the graph convolutional attention layer (CAT), which relies on convolutions to compute the attention scores. Unfortunately, as in the case of GCNs and GATs, we show that there exists no clear winner between the three—neither theoretically nor in practice—as their performance directly depends on the nature of the data (i.e., of the graph and features). This result brings us to the main contribution of our work, the learnable graph convolutional attention network (L-CAT): a GNN architecture that automatically interpolates between GCN, GAT and CAT in each layer, by adding two scalar parameters. Our results demonstrate that L-CAT is able to efficiently combine different GNN layers along the network, outperforming competing methods in a wide range of datasets, and resulting in a more robust model that reduces the need of cross-validating.

1 Introduction

In recent years, Graph Neural Networks (GNNs) (Scarselli et al., 2008) have become ubiquitous in machine learning, emerging as the standard approach in many settings. For example, they have been successfully applied for tasks such as topic prediction in citation networks (Sen et al., 2008); molecule prediction (Gilmer et al., 2017); and link prediction in recommender systems (Wu et al., 2020a). These applications typically make use of message-passing GNNs (Gilmer et al., 2017), whose idea is fairly simple: in each layer, nodes are updated by aggregating the information (messages) coming from their neighboring nodes.

Depending on how this aggregation is implemented, we can define different types of GNN layers. Two important and widely adopted layers are graph convolutional networks (GCNs) (Kipf & Welling, 2017), which uniformly average the neighboring information; and graph attention networks (GATs) (Velickovic et al., 2018), which instead perform a weighted average, based on an attention score between receiver and sender nodes. More recently, a number of works have shown the strengths and limitations of both approaches from a theoretical (Fountoulakis et al., 2022; Baranwal et al., 2021; 2022), and empirical (Knyazev et al., 2019) point of view. These results show that their performance depends on the nature of the data at hand (i.e., the graph and the features), thus the standard approach is to select between GCNs and GATs via computationally demanding cross-validation.

In this work, we aim to exploit the benefits of both convolution and attention operations in the design of GNN architectures. To this end, we first introduce a novel graph convolutional attention layer (CAT), which extends existing attention layers by taking the convolved

*Equal contribution. Correspondence to: {ajavaloy,sanchez}@cs.uni-saarland.de.

features as inputs of the score function. Following (Fountoulakis et al., 2022), we rely on a contextual stochastic block model to theoretically compare GCN, GAT, and CAT architectures. Our analysis shows that, unfortunately, no free lunch exists among these three GNN architectures since their performance, as expected, is fully data-dependent.

This motivates the main contribution of the paper, the *learnable graph convolutional attention network* (L-CAT): a novel GNN which, in each layer, automatically interpolates between the three operations by introducing only two scalar parameters. As a result, L-CAT is able to learn the proper operation to apply at each layer, thus combining different layer types in the same GNN architecture while overcoming the need to cross-validate—a process that was prohibitively expensive prior to this work. Our extensive empirical analysis demonstrates the capabilities of L-CAT on a wide range of datasets, outperforming existing baseline GNNs in terms of both performance, and robustness to input noise and network initialization.

2 Preliminaries

Assume as input an undirected graph $G = (V, E)$, where $V = [n]$ denotes the set of vertices of the graph, and $E \subseteq V \times V$ the set of edges. Each node $i \in [n]$ is represented by a d-dimensional feature vector $\mathbf{X}_i \in \mathbb{R}^d$, and the goal is to produce a set of predictions $\{\hat{\boldsymbol{y}}_i\}_{i=1}^n$. To this end, a message-passing GNN layer yields a representation $\tilde{\boldsymbol{h}}_i \in \mathbb{R}^{d'}$ for each node i, by collecting and aggregating the information from each of its neighbors into a single message; and using the aggregated message to update its representation from the previous layer, $\boldsymbol{h}_i \in \mathbb{R}^d$. For the purposes of this work, we can define this operation as the following:

$$\tilde{\boldsymbol{h}}_i = f(\boldsymbol{h}'_i) \quad \text{where} \quad \boldsymbol{h}'_i \stackrel{\text{def}}{=} \sum_{j \in N_i^*} \gamma_{ij} \boldsymbol{W}_v \boldsymbol{h}_j \ , \tag{1}$$

where N_i^* is the set of neighbors of node i (including i), $\boldsymbol{W}_v \in \mathbb{R}^{d' \times d}$ a learnable matrix, f an elementwise function, and $\gamma_{ij} \in [0, 1]$ are coefficients such that $\sum_j \gamma_{ij} = 1$ for each node i.

Let the input features be $\boldsymbol{h}_i^0 = \mathbf{X}_i$, and $\boldsymbol{h}_i^L = \hat{\boldsymbol{y}}_i$ the predictions, then we can readily define a message-passing GNN (Gilmer et al., 2017) as a sequence of L layers as defined above. Depending on the way the coefficients γ_{ij} are computed, we identify different GNN flavors.

Graph convolutional networks (GCNs) (Kipf & Welling, 2017) are simple yet effective. In short, GCNs compute the average of the messages, i.e., they assign the same coefficient $\gamma_{ij} = 1/|N_i^*|$ to every neighbor:

$$\tilde{\boldsymbol{h}}_i = f(\boldsymbol{h}'_i) \quad \text{where} \quad \boldsymbol{h}'_i \stackrel{\text{def}}{=} \frac{1}{|N_i^*|} \sum_{j \in N_i^*} \boldsymbol{W}_v \boldsymbol{h}_j \ , \tag{2}$$

Graph attention networks take a different approach. Instead of assigning a fixed value to each coefficient γ_{ij}, they compute it as a function of the sender and receiver nodes. A general formulation for these models can be written as follows:

$$\tilde{\boldsymbol{h}}_i = f(\boldsymbol{h}'_i) \quad \text{where} \quad \boldsymbol{h}'_i \stackrel{\text{def}}{=} \sum_{j \in N_i^*} \gamma_{ij} \boldsymbol{W}_v \boldsymbol{h}_j \quad \text{and} \quad \gamma_{ij} \stackrel{\text{def}}{=} \frac{\exp(\Psi(\boldsymbol{h}_i, \boldsymbol{h}_j))}{\sum_{\ell \in N_i^*} \exp(\Psi(\boldsymbol{h}_i, \boldsymbol{h}_\ell))} \ . \tag{3}$$

Here, $\Psi(\boldsymbol{h}_i, \boldsymbol{h}_j) \stackrel{\text{def}}{=} \alpha(\boldsymbol{W}_q \boldsymbol{h}_i, \boldsymbol{W}_k \boldsymbol{h}_j)$ is known as the *score function* (or *attention architecture*), and provides a score value between the messages $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ (or more generally, between a learnable mapping of the messages). From these scores, the (attention) coefficients are obtained by normalizing them, such that $\sum_j \gamma_{ij} = 1$. We can find in the literature different attention layers and, throughout this work, we focus on the original GAT (Velickovic et al., 2018) and its extension GATv2 (Brody et al., 2022):

$$\text{GAT:} \quad \Psi(\boldsymbol{h}_i, \boldsymbol{h}_j) = \text{LeakyRelu}\left(\boldsymbol{a}^\top [\boldsymbol{W}_q \boldsymbol{h}_i || \boldsymbol{W}_k \boldsymbol{h}_j]\right) \ , \tag{4}$$

$$\text{GATv2:} \quad \Psi(\boldsymbol{h}_i, \boldsymbol{h}_j) = \boldsymbol{a}^\top \text{LeakyRelu}\left(\boldsymbol{W}_q \boldsymbol{h}_i + \boldsymbol{W}_k \boldsymbol{h}_j\right) \ , \tag{5}$$

where the learnable parameters are now the attention vector $\boldsymbol{a}$; and the matrices $\boldsymbol{W}_q$, $\boldsymbol{W}_k$, and $\boldsymbol{W}_v$. Following previous work (Velickovic et al., 2018; Brody et al., 2022), we assume that these matrices are coupled, i.e., $\boldsymbol{W}_q = \boldsymbol{W}_k = \boldsymbol{W}_v$. Note that the difference between the

two layers lies in the position of the vector $\boldsymbol{a}$: by taking it out of the nonlinearity, Brody et al. (2022) increased the expressiveness of GATv2. Now, the product of $\boldsymbol{a}$ and a weight matrix does not collapse into another vector. More importantly, the addition of two different attention layers will help us show the versatility of the proposed models later in §6.

3 Previous work

In recent years, there has been a surge of research in GNNs. Here, we discuss other GNN models, attention mechanisms, and the recent findings on the limitations of GCNs and GATs.

The literature on GNNs is extensive (Wu et al., 2020b; Hamilton et al., 2017a; Battaglia et al., 2018; Lee et al., 2019), and more abstract definitions of a message-passing GNN are possible, leading to other lines of work trying different ways to compute messages, aggregate them, or update the final message (Hamilton et al., 2017b; Xu et al., 2019; Corso et al., 2020). Alternatively, another line of work fully abandons message-passing, working instead with higher-order interactions (Morris et al., 2019). While some of this work is orthogonal—or directly applicable—to the proposed model, in the main paper we focus on convolutional and attention graph layers, as they are the most widely used (and cited) as of today.

While we consider the original GAT (Velickovic et al., 2018) and GATv2 (Brody et al., 2022), our work can be directly applied to any attention model that sticks to the formulation in Eq. 3. For example, some works propose different metrics for the score function, like the dot-product (Brody et al., 2022), cosine similarity (Thekumparampil et al., 2018), or a combination of various functions (Kim & Oh, 2021). Other works introduce transformer-based mechanisms (Vaswani et al., 2017) based on positional encoding (Dwivedi & Bresson, 2020; Kreuzer et al., 2021) or on the set transformer (Wang et al., 2021a). Finally, there also exist attention approaches designed for specific type of graphs, such as relational (Yun et al., 2019; Busbridge et al., 2019) or heterogeneous graphs (Wang et al., 2019b; Hu et al., 2020b).

3.1 On the limitations of GCN and GAT networks

Baranwal et al. (2021) studied classification on a simple stochastic block model, showing that, when the graph is neither too sparse nor noisy, applying one layer of graph convolution increases the regime in which the data is linearly separable. However, this result is highly sensitive to the graph structure, as convolutions essentially collapse the data to the same value in the presence of enough noise. More recently, Fountoulakis et al. (2022) showed that GAT is able to remedy the above issue, and provides perfect node separability regardless of the noise level in the graph. However, a classical argument (see Anderson (2003)) states that *in this particular setting* a linear classifier already achieves perfect separability. These works, in summary, showed scenarios for which GCNs can be beneficial in the absence of noise, and that GAT can outperform GCNs in other scenarios, leaving open the question of which architecture is preferable in terms of performance.

4 Convolved attention: benefits and hurdles

In this section, we propose to combine attention with convolution operations. To motivate it, we complement the results of Fountoulakis et al. (2022), providing a synthetic dataset for which *any* 1-layer GCN fails, but 1-layer GAT does not. Thus, proving a clear distinction between GAT and GCN layers. Besides, we show that convolution helps GAT as long as the graph noise is reasonable. The proofs for the two statements in this section appear in Appendix A and follow similar arguments as in Fountoulakis et al. (2022).

This dataset is based on the *contextual stochastic block model* (CSBM) (Deshpande et al., 2018). Let $\varepsilon_1, \dots, \varepsilon_n$ be iid. uniform samples from $\{-1, 0, 1\}$. Let $C_k = \{j \in [n] \mid \varepsilon_j = k\}$ for $k \in \{-1, 0, 1\}$. We set the feature vector $\mathbf{X}_i \sim \mathcal{N}(\varepsilon_i \boldsymbol{\mu}, \mathbf{I} \cdot \sigma^2)$ where $\boldsymbol{\mu} \in \mathbb{R}^d$, $\sigma \in \mathbb{R}$, and $\mathbf{I} \in \{0, 1\}^{d \times d}$ is the identity matrix. For a given pair $p, q \in [0, 1]$ we consider the stochastic adjacency matrix $\mathbf{A} \in \{0, 1\}^{n \times n}$ defined as follows: for $i, j \in [n]$ in the same class (*intra-edge*), we set $a_{ij} \sim \text{Ber}(p)$;[1] for i, j in different classes (*inter-edge*), we set $a_{ij} \sim \text{Ber}(q)$. We denote

[1] $\text{Ber}(\cdot)$ denote the Bernoulli distribution.

by $(\mathbf{X}, \mathbf{A}) \sim \mathsf{CSBM}(n, p, q, \boldsymbol{\mu}, \sigma^2)$ a sample obtained according to the above random process. Our task is then to distinguish (or separate) nodes from C_0 vs. $C_{-1} \cup C_1$.

Note that, in general, it is impossible to separate C_0 from $C_{-1} \cup C_1$ with a linear classifier and, using one convolutional layer is detrimental for node classification on the CSBM:[2] although the convolution brings the means closer and shrinks the variance, the geometric structure of the problem does not change. On the other hand, we prove that GAT is able to achieve perfect node separability when the graph is not too sparse:

Theorem 1. Suppose that $p, q = \Omega(\log^2 n/n)$ and $\|\boldsymbol{\mu}\|_2 = \omega(\sigma\sqrt{\log n})$. Then, there exists a choice of attention architecture Ψ such that, with probability at least $1 - o_n(1)$ over the data $(\mathbf{X}, \mathbf{A}) \sim \mathsf{CSBM}(n, p, q, \boldsymbol{\mu}, \sigma^2)$, GAT separates nodes C_0 from $C_1 \cup C_{-1}$.

Moreover, we show using methods from Baranwal et al. (2021), that the above classification threshold $\|\boldsymbol{\mu}\|$ can be improved when the graph noise is reasonable. Specifically, *by applying convolution prior to the attention score*, the variance of the data is greatly reduced, and if the graph is not too noisy, the operation dramatically lowers the bound in Thm. 1. We exploit this insight by introducing the *graph convolutional attention layer* (CAT):

$$\Psi(\boldsymbol{h}_i, \boldsymbol{h}_j) = \alpha(\boldsymbol{W}\tilde{\boldsymbol{h}}_i, \boldsymbol{W}\tilde{\boldsymbol{h}}_j) \quad \text{where} \quad \tilde{\boldsymbol{h}}_i = \frac{1}{|N_i^*|} \sum_{\ell \in N_i^*} \boldsymbol{h}_\ell \,, \tag{6}$$

where $\tilde{\boldsymbol{h}}_i$ are the convolved features of the neighborhood of node i. As we show now, CAT improves over GAT by combining convolutions with attention, when the graph noise is low.

Corollary 2. Suppose $p, q = \Omega(\log^2 n/n)$ and $\|\boldsymbol{\mu}\| \geq \omega\left(\sigma\sqrt{\frac{(p+2q)\log n}{n(p-q)^2}}\right)$. Then, there is a choice of attention architecture Ψ such that CAT separates nodes C_0 from $C_1 \cup C_{-1}$, with probability at least $1 - o(1)$ over the data $(\mathbf{X}, \mathbf{A}) \sim \mathsf{CSBM}(n, p, q, \boldsymbol{\mu}, \sigma^2)$.

The above proposition shows that under the CSBM data model, convolving prior to attention changes the regime for perfect node separability by a factor of $|p-q|\sqrt{n/(p+2q)}$. This is desirable when $|p-q|\sqrt{n/(p+2q)} > 1$, since the regime for perfect classification is increased. Otherwise, applying convolution prior to attention reduces the regime for perfect separability. Therefore, it is not always clear whether convolving prior to attention is beneficial.

5 L-CAT: Learning to interpolate

From the previous analysis, we can conclude that it is hard to know *a priori* whether attention, convolution, or convolved attention, will perform the best. In this section, we argue that this issue can be easily overcome by learning to interpolate between the three.

First, note that GCN and GAT only differ in that GCN weighs all neighbors equally (Eq. 2) and, the more similar the attention scores are (Eq. 3), the more uniform the coefficients γ_{ij} are. Thus, we can interpolate between GCN and GAT by introducing a learnable parameter. Similarly, the formulation of GAT (Eq. 3) and CAT (Eq. 6) differ in the convolution within the score, which can be interpolated with another learnable parameter.

Following this observation, we propose the *learnable convolutional attention layer* (L-CAT), which can be formulated as an attention layer with the following score:

$$\Psi(\boldsymbol{h}_i, \boldsymbol{h}_j) = \lambda_1 \cdot \alpha(\boldsymbol{W}\tilde{\boldsymbol{h}}_i, \boldsymbol{W}\tilde{\boldsymbol{h}}_j) \quad \text{where} \quad \tilde{\boldsymbol{h}}_i = \frac{\boldsymbol{h}_i + \lambda_2 \sum_{\ell \in N_i} \boldsymbol{h}_\ell}{1 + \lambda_2 |N_i|} \,, \tag{7}$$

where $\lambda_1, \lambda_2 \in [0, 1]$. As mentioned before, this formulation lets L-CAT learn to interpolate between GCN ($\lambda_1 = 0$), GAT ($\lambda_1 = 1$ and $\lambda_2 = 0$), and CAT ($\lambda_1 = 1$ and $\lambda_2 = 1$).

L-CAT enables a number of non-trivial benefits. Not only can it switch between existing layers, but it also learns the amount of attention necessary for each use-case. Moreover, by

[2] We note that this problem can be easily solved by two layers of GCN (Baranwal et al., 2022).

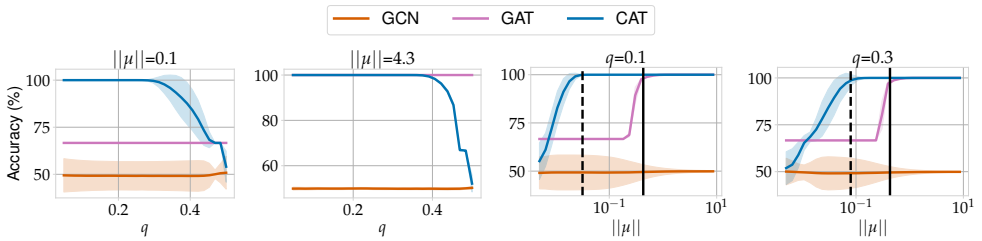


Figure 1: Synthetic data results. The left-most plots show accuracy as we vary the noise level q for $\|\boldsymbol{\mu}\| = 0.1$ and $\|\boldsymbol{\mu}\| = 4.3$. The right-most plots show the accuracy as we change the norm of the means $\|\boldsymbol{\mu}\|$ for $q = 0.1$ and $q = 0.3$. We use two vertical lines to present the classification threshold stated in Thm. 1 (solid line) and Cor. 2 (dashed line).

comprising the three layers in a single learnable formulation, it removes the necessity of cross-validating the type of layer, as their performance is data-dependent (see §§3.1 and 4). Remarkably, it allows to easily combine different layer types within the same architecture.

While we focus on GCN, L-CAT can be easily used with other GNN architectures such as PNA (Corso et al., 2020) and GCNII (Chen et al., 2020), as L-CAT interpolates between two different adjacency matrices. For further details and results, refer to App. F.

6 Experiments

In this section, we first validate our theoretical findings on synthetic data (§6.1). Then, we show through various node classification tasks that (L-)CAT is as competitive as the baseline models (§6.2). Lastly, we move to more demanding scenarios from the Open Graph Benchmark (Hu et al., 2020a), demonstrating that L-CAT is a more flexible and robust alternative to its baseline methods (§6.3) that reduces the need of cross-validating without giving up on performance. Refer to Apps. B to F for details and additional results. The code to reproduce the experiments can be found at `https://github.com/psanch21/LCAT`.

6.1 Synthetic data

First, we empirically validate our theoretical results (Thm. 1 and Cor. 2). We aim to better understand the behavior of each layer as the properties of the data change, i.e., the noise level q (proportion of inter-edges) and the distance between the means of consecutive classes $\|\boldsymbol{\mu}\|$. We provide extra results and additional experiments in App. B.

Experimental setup. As data model, we use the proposed CSBM (see §4) with $n = 10000$, $p = 0.5$, $\sigma = 0.1$, and $d = n/\left(5\log^2(n)\right)$. All results are averaged over 50 runs, and parameters are set as described in App. A. We conduct two experiments to assess the sensitivity to structural noise. First, we vary the noise level q between 0 and 0.5, leaving the mean vector $\boldsymbol{\mu}$ fixed. We test two values of $\|\boldsymbol{\mu}\|$: the first corresponds to the *easy* regime ($\|\boldsymbol{\mu}\| = 10\sigma\sqrt{2\log n}$) where classes are far apart; and the second correspond to the *hard* regime ($\|\boldsymbol{\mu}\| = \sigma$) where clusters are close. In the second experiment we instead sweep $\|\boldsymbol{\mu}\|$ in the range $\left[\sigma/20, 20\sigma\sqrt{2\log n}\right]$, covering the transition from hard (small $\|\boldsymbol{\mu}\|$) to easy (large $\|\boldsymbol{\mu}\|$) settings. Here, we fix q to 0.1 (low noise) and 0.3 (high noise). In both cases, we compare the behavior of 1-layer GAT and CAT, and include GCN as the baseline.

Results. The two left-most plots of Fig. 1 show node classification performance for the hard and easy regimes, respectively, as we vary the noise level q. In the hard regime, we observe that GAT is unable to achieve separation for any value of q, whereas CAT achieves perfect classification when q is small enough. This exemplifies the advantage of CAT over GAT as stated in Cor. 2. When the distance between the means is large enough, we see that GAT achieves perfect results independently of q, as stated in Thm. 1. In contrast, when CAT fails to satisfy the condition in Cor. 2 (as we increase q), it achieves inferior performance.

The right-most part of Fig. 1 shows the results when we fix q and sweep $\|\boldsymbol{\mu}\|$. In these two plots, we can appreciate the transition in the accuracy of both GAT and CAT as a function of $\|\boldsymbol{\mu}\|$. We observe that GAT achieves perfect accuracy when the distance between the means satisfies the condition in Thm. 1 (solid vertical line in Fig. 1). Moreover, we can see

Table 1: Test accuracy (%) of the considered models for different datasets (sorted by their average node degree), and averaged over ten runs. Bold numbers are statistically different to their baseline model ($\alpha = 0.05$). Best average performance is underlined.

Dataset	*Amazon Computers*	*Amazon Photo*	*GitHub*	*Facebook PagePage*	*Coauthor Physics*	*TwitchEN*
Avg. Deg.	35.76	31.13	15.33	15.22	14.38	10.91
GCN	90.59 ± 0.36	95.13 ± 0.57	84.13 ± 0.44	94.76 ± 0.19	96.36 ± 0.10	57.83 ± 1.13
GAT	89.59 ± 0.61	94.02 ± 0.66	83.31 ± 0.18	94.16 ± 0.48	96.36 ± 0.10	57.59 ± 1.20
CAT	**90.58 ± 0.40**	**94.77 ± 0.47**	**84.11 ± 0.66**	**94.71 ± 0.30**	96.40 ± 0.10	58.09 ± 1.61
L-CAT	**90.34 ± 0.47**	**94.93 ± 0.37**	**84.05 ± 0.70**	**94.81 ± 0.25**	96.35 ± 0.10	57.88 ± 2.07
GATv2	89.49 ± 0.53	93.47 ± 0.62	82.92 ± 0.45	93.44 ± 0.30	96.24 ± 0.19	57.70 ± 1.17
CATv2	**90.44 ± 0.46**	**94.81 ± 0.55**	**84.10 ± 0.88**	**94.27 ± 0.31**	96.34 ± 0.12	57.99 ± 2.02
L-CATv2	**90.33 ± 0.44**	**94.79 ± 0.61**	**84.31 ± 0.59**	**94.44 ± 0.39**	96.29 ± 0.13	57.89 ± 1.53

the improvement CAT obtains over GAT. Indeed, when $\|\boldsymbol{\mu}\|$ satisfies the conditions of Cor. 2 (dashed vertical line in Fig. 1), the classification threshold is improved. As we increase q we see that the gap between the two vertical lines decreases, which means that the improvement of CAT over GAT decreases as q increments, exactly as stated in Cor. 2.

6.2 Real data

We study now the performance of the proposed models in a comprehensive set of real-world experiments, in order to gain further insights of the settings in which they excel. Specifically, we found CAT and L-CAT to outperform their baselines as the average node degree increases. For a detailed description of the datasets and additional results, refer to Apps. C, D and F.

Models. We consider as baselines a simple GCN layer (Kipf & Welling, 2017), the original GAT layer (Velickovic et al., 2018) and its recent extension, GATv2 (Brody et al., 2022). Based on the two attention models, we consider their CAT and L-CAT extensions. To ensure fair comparisons, all layers use the same number of parameters and implementation.

Datasets. We consider six node classification datasets. The *Facebook*/*GitHub*/*TwitchEN* datasets involve social-network graphs (Rozemberczki et al., 2021), whose nodes represent verified pages/developers/streamers; and where the task is to predict the topic/expertise/explicit-language-use of the node. The *Coauthor Physics* dataset (Shchur et al., 2018) represents a co-authorship network whose nodes represent authors, and the task is to infer their main research field. The *Amazon* datasets represent two product-similarity graphs (Shchur et al., 2018), where each node is a product, and the task is to infer its category.

Experimental setup. To ensure the best results, we cross-validate all optimization-related hyperparameters for each model using GraphGym (You et al., 2020). All models use four GNN layers with hidden size of 32, and thus have an equal number of parameters. For evaluation, we take the best-validation configuration during training, and report test-set performance. For further details, refer to App. D.

Results are presented in Table 1. In contrast with §6.1, we here find GCN to be a strong contender, reinforcing its viability in real-world data despite its simplicity. We observe both CAT and L-CAT not only holding up the performance with respect to their baselines models for all datasets, but in most cases also improving the test accuracy in a statistically significant manner. These results validate the effectiveness of CAT as a GNN layer, and show the viability of *L-CAT as a drop-in replacement*, achieving good results on all datasets.

As explained in §4, CAT differs from a usual GAT in that the score is computed with respect to the convolved features. Intuitively, this means that CAT should excel in those settings where nodes are better connected, allowing CAT to extract more information from their neighborhoods. Indeed, in the inset figure we can observe the improvement in accuracy of CAT with respect to its baseline model, as a function of the average node degree of the dataset, and the linear regression fit of these results (dashed line). This plot includes all datasets

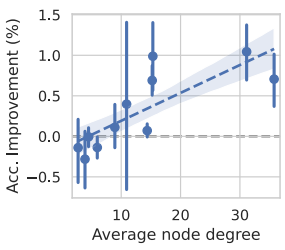


Table 2: Test performance of the considered models on four OGB datasets, averaged over five runs. Bold numbers are statistically different to their baseline model ($\alpha = 0.05$). Best average performance is underlined. Left table: accuracy (%); right table: AUC-ROC (%).

Dataset	*arxiv*	*products*	*mag*	*proteins*
GCN	71.58 ± 0.20	74.12 ± 1.20	32.77 ± 0.36	80.10 ± 0.55
GAT	71.58 ± 0.16	78.53 ± 0.91	32.15 ± 0.31	79.08 ± 1.47
CAT	**72.14 ± 0.21**	**77.38 ± 0.36**	31.98 ± 0.46	73.26 ± 1.65
L-CAT	**71.99 ± 0.08**	77.19 ± 1.11	32.47 ± 0.38	79.63 ± 0.71
GATv2	71.73 ± 0.24	76.40 ± 0.71	32.76 ± 0.18	78.65 ± 1.44
CATv2	**72.03 ± 0.09**	74.81 ± 1.12	**32.43 ± 0.22**	74.33 ± 0.94
L-CATv2	71.97 ± 0.22	**76.37 ± 0.92**	32.68 ± 0.50	**79.07 ± 0.98**

(from the manuscript and App. D), and shows a positive trend between node connectivity and improved performance achieved by CAT.

6.3 Open Graph Benchmark

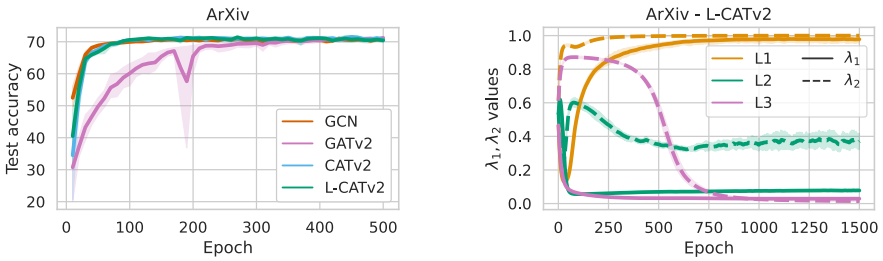
In this section, we assess the robustness of the proposed models, in order to fully understand their benefits. For further details and additional results, refer to App. E.

Datasets. We consider four datasets from the OGB suite (Hu et al., 2020a): *proteins*, *products*, *arxiv*, and *mag*. Note that these datasets are significantly larger than those from §6.2 and correspond to more difficult tasks, e.g., *arxiv* is a 40-class classification problem (see Table 5 in App. C for details). This makes them more suitable for the proposed analysis.

Experimental setup. We adopt the same experimental setup as Brody et al. (2022) for the *proteins*, *products*, and *mag* datasets. For the *arxiv* dataset, we use instead the example code from OGB (Hu et al., 2020a), as it yields better performance than that of Brody et al. (2022). Just as in §6.2, we compare with GCN (Kipf & Welling, 2017), GAT (Velickovic et al., 2018), GATv2 (Brody et al., 2022), and their CAT and L-CAT counterparts. We cross-validate the number of heads (1 and 8), and select the best-validation models during training. All models are identical except for their λ values.

Results are summarized in Table 2. Here we do not observe a clear preferred baseline: GCN performs really well in *proteins* and *mag*; GAT excels in *products*; and GATv2 does well in *arxiv* and *mag*. While CAT obtains the best results on *arxiv*, its performance on *proteins* and *products* is significantly worse than the baseline model. Presumably, an excessive amount of inter-edges could explain why convolving the features prior to computing the score is harmful, as seen in §6.1. As we explore in §6.3.2, however, CAT improves over its baseline for most *proteins* scenarios, specially with a single head. In stark contrast, L-CAT performs remarkably well, improving the baseline models in all datasets but *products*—even on those in which CAT fails—demonstrating the adaptability of L-CAT to different scenarios.

To better understand the training dynamics of the models, we plot in Fig. 2a the test accuracy of GCN and the GATv2 models during training on the *arxiv* dataset. Interestingly, despite all models obtaining similar final results, *CATv2 and L-CATv2 drastically improved their convergence speed and stability with respect to GATv2*, matching that of GCN. To understand the behavior of L-CATv2, Fig. 2b shows the evolution of the λ parameters. We observe that, to achieve these results, L-CATv2 converged to a GNN network that combines three types of layers: the first layer is a CATv2 layer, taking advantage of the neighboring information; the second layer is a quasi-GCN layer, in which scores are almost uniform and some neighboring information is still used in the score computation; and the third layer is a pure GCN layer, in which all scores are uniformly distributed. It is important to remark that these dynamics are fairly consistent, as L-CATv2 reached the same λ values over all five runs.



(a) Test accuracy.

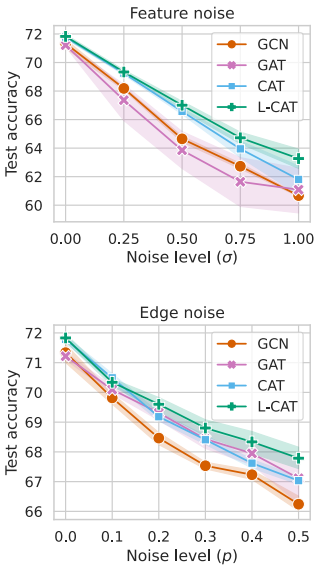
(b) Evolution of λ_1, λ_2.

Figure 2: Behavior of GCN and GATv2-based models during training on the *arxiv* dataset. *(a)* CAT and L-CAT converge quicker and more stably than their baseline model. *(b)* L-CAT consistently converges to the same architecture: a CAT →quasi-GCN→GCN network.

6.3.1 Robustness to noise

One intrinsic aspect of real world data is the existence of noise. In this section, we explore the robustness of the proposed models to different levels of noise, i.e., we attempt to simulate scenarios where there exist measurement inaccuracies in the input features and edges.

Experimental setup. We consider the *arxiv* dataset, and the same experimental setup as in §6.3. We conduct two experiments. First, we introduce to the node features additive noise of the form $\mathcal{N}(\mathbf{0}, \mathbf{1}\sigma)$, and consider different levels of noise, $\sigma \in \{0, 0.25, 0.5, 0.75, 1\}$. Then, as in (Brody et al., 2022), we simulate edge noise by adding fake edges with probability $Bern(p)$ for $p \in \{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$.



Results are shown in the inset figures for feature (top) and edge noise (bottom), summarizing the performance of all models over five runs and two numbers of heads (1 and 8). Baseline attention models are quite sensitive to feature noise, but are more robust to edge noise, as they can drop inter-class edges (see §3.1). GCNs, as expected, are instead more robust to feature noise, but suffer more in the presence of edge noise. In concordance with the synthetic experiments (see §§4 and 6.1), CAT is able to leverage convolutions as a variance-reduction technique, reducing the variance and improving its robustness to feature noise. Remarkably, L-CAT proves to be the most robust for both types of noise: by adapting the amount of attention used in each layer, it outperforms existing methods and reduces the variance.

6.3.2 Robustness to network initialization

Another important aspect for real-world applications is robustness to network initialization, i.e., the ability to obtain satisfying performance independently of the initial parameters. Otherwise, a practitioner can waste lots of resources trying initilizations or, even worse, give up on a model just because they did not try the initial parameters that yield great results.

Experimental setup. We follow once again the same setup for *proteins* as in §6.3. We consider two different network initializations. The first one, *uniform*, uses uniform Glorot initilization (Glorot & Bengio, 2010) with a gain of 1, which is the standard initialization used throughout this work. The second one, *normal*, uses instead normal Glorot initialization (Glorot & Bengio, 2010) with a gain of $\sqrt{2}$. This is the initialization employed on the original GATv2 paper (Brody et al., 2022) exclusively for the *proteins* dataset.

Results—segregated by number of heads—are shown in Table 3, while the results for GCN appear in the inset table. These results show that the baseline models perform poorly on the

Table 3: Test AUC-ROC (%) on the *proteins* dataset for attention models with two different network initializations (see §6.3.2), using 1 head (top) and 8 heads (bottom).

		GAT	CAT	L-CAT	GATv2	CATv2	L-CATv2
1h	*uniform*	59.73 ± 3.61	**64.32 ± 2.33**	**77.77 ± 1.28**	59.85 ± 2.73	**64.32 ± 2.33**	**79.08 ± 0.95**
	normal	66.38 ± 6.94	73.26 ± 1.65	**78.06 ± 1.25**	69.13 ± 8.48	74.33 ± 0.94	**79.07 ± 0.98**
8h	*uniform*	72.23 ± 2.86	73.60 ± 1.14	**78.85 ± 1.57**	75.21 ± 1.61	74.16 ± 1.30	**78.77 ± 0.97**
	normal	79.08 ± 1.47	**74.67 ± 1.15**	79.63 ± 0.71	78.65 ± 1.44	**73.40 ± 0.56**	79.30 ± 0.49
	average	69.36 ± 8.52	73.93 ± 1.35	78.58 ± 1.48	70.71 ± 8.70	71.55 ± 4.54	79.05 ± 0.91

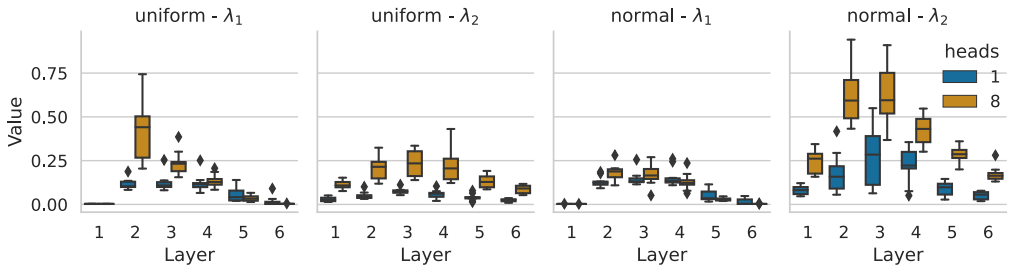


Figure 3: Distribution of λ_1, λ_2 on *proteins* dataset for L-CAT across initializations.

uniform initialization. However, this is somewhat alleviated when using 8 heads in the attention models. Moreover, all baselines significantly improve with *normal* initialization, being GCN the best model, and attention models obtaining 79 % accuracy on average with 8 heads. Compared to the baselines, CAT does a good job and improves the performance in all cases except for *normal* with 8 heads. Remarkably, L-CAT consistently obtains high accuracy in all scenarios and runs. To emphasize consistency, bottom row shows the average accuracy across runs, showing that L-CAT is clearly more robust to parameter initialization than competing models.

	GCN
uniform	61.08 ± 2.56
normal	80.10 ± 0.55
average	70.59 ± 10.21

To understand this performance, we inspect the distribution of λ_1, λ_2 for L-CAT in Fig. 3. Here, we can spot a few interesting patterns. Consistently, the first and last layers are always GCNs, while the inner layers progressively admit less attention. Second, the number of heads affects the amount of attention allowed in the network; the more heads, the more expressive the layer tends to be, and more attention is permitted. Third, L-CAT adapts to the initialization used: in *uniform*, it allows more attention in the second layer; in *normal*, it allows more attention in the score inputs. These results consolidate the flexibility of L-CAT.

7 Conclusions and future work

In this work, we studied how to combine the strengths of convolution and attention layers in GNNs. We proposed CAT, which computes attention with respect to the convolved features, and analyzed its benefits and limitations on a new synthetic dataset. This analysis revealed different regimes where one model is preferred over the others, reinforcing the idea that selecting between GCNs, GATs, and now CATs, is a difficult task. For this reason, we proposed L-CAT, a model which interpolates between the three via two learnable parameters. Extensive experimental results demonstrated the effectiveness of L-CAT, yielding great results while being more robust than other methods. As a result, L-CAT proved to be a viable drop-in replacement that removes the need to cross-validate the layer type.

We strongly believe learnable interpolation can get us a long way, and we hope L-CAT to motivate new and exciting work. Specially, we are eager to see L-CAT in real applications, and thus finding out what combining different GNN layers across a model (without the hurdle of cross-validating all layer combinations) can lead to in the real-world.

Ethic statement

Given the nature of this work, we do not see any direct ethical concerns. On the contrary, L-CAT eases the application of GNNs to the practitioner, and removes the need of cross-validating the layer type, which can potentially benefit other areas and applications, as GNNs have already proven.

Reproducibility statement

For the theoretical results, we describe the data model used in §4, and provide all the detailed proofs in App. A. For the experimental results, we include in the supplementary material the necessary code and scripts required to reproduce our experiments, and all required datasets are freely available. Complete details about the experimental setup can be found in Apps. B, D and E, and we report in our results the mean and standard deviation computed using five trials or more. In addition, we highlight in bold the results that are statistically significant. Moreover, we include details of computational resources used for the all sets of experiments in App. B, App. D and App. E.

Acknowledgements

We would like to thank Batuhan Koyuncu, Jonas Klesen, Miriam Rateike, and Maryam Meghdadi for helpful feedback and discussions. Pablo Sánchez Martín thanks the German Research Foundation through the Cluster of Excellence "Machine Learning – New Perspectives for Science", EXC 2064/1, project number 390727645 for generous funding support. The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Pablo Sánchez Martín.

References

T.W. Anderson. *An introduction to multivariate statistical analysis.* John Wiley & Sons, 2003. (page 3)

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. URL https://arxiv.org/abs/1607.06450. (page 30)

Aseem Baranwal, Kimon Fountoulakis, and Aukosh Jagannath. Graph convolution for semi-supervised classification: Improved linear separability and out-of-distribution generalization. In *International Conference on Machine Learning (ICML)*. PMLR, 2021. (page 1, 3, 4, 16)

Aseem Baranwal, Kimon Fountoulakis, and Aukosh Jagannath. Effects of graph convolutions in deep networks. *arXiv preprint arXiv:2204.09297*, 2022. URL https://arxiv.org/abs/2204.09297. (page 1, 4)

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018. URL https://arxiv.org/abs/1806.01261. (page 3)

K. Bhatia, K. Dahiya, H. Jain, P. Kar, A. Mittal, Y. Prabhu, and M. Varma. The extreme classification repository: Multi-label datasets and code, 2016. URL http://manikvarma.org/downloads/XC/XMLRepository.html. (page 27)

Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=r1ZdKJ-0W. (page 27)

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *International Conference on Learning Representations (ICLR)*, 2022. (page 2, 3, 6, 7, 8, 30, 31)

Dan Busbridge, Dane Sherburn, Pietro Cavallo, and Nils Y Hammerla. Relational graph attention networks. *arXiv preprint arXiv:1904.05811*, 2019. URL https://arxiv.org/abs/1904.05811. (page 3)

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1725–1735. PMLR, 2020. URL http://proceedings.mlr.press/v119/chen20v.html. (page 5, 32, 34, 35)

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Velickovic. Principal neighbourhood aggregation for graph nets. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/99cad265a1768cc2dd013f0e740300ae-Abstract.html. (page 3, 5, 32, 33, 34)

Yash Deshpande, Subhabrata Sen, Andrea Montanari, and Elchanan Mossel. Contextual stochastic block models. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 8590–8602, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/08fc80de8121419136e443a70489c123-Abstract.html. (page 3)

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020. URL https://arxiv.org/abs/2012.09699. (page 3)

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019. (page 28, 29)

Kimon Fountoulakis, Amit Levi, Shenghao Yang, Aseem Baranwal, and Aukosh Jagannath. Graph attention retrospective. *arXiv preprint arXiv:2202.13060*, 2022. URL https://arxiv.org/abs/2202.13060. (page 1, 2, 3, 16, 18, 20)

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1263–1272. PMLR, 2017. URL http://proceedings.mlr.press/v70/gilmer17a.html. (page 1, 2)

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010. (page 8)

William L. Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *IEEE Data Eng. Bull.*, 40, 2017a. (page 3)

William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 1024–1034, 2017b. URL https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html. (page 3)

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pp. 1026–1034. IEEE Computer Society, 2015. doi: 10.1109/ICCV.2015.123. URL https://doi.org/10.1109/ICCV.2015.123. (page 28)

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020a. URL https://proceedings.neurips.cc/paper/2020/hash/fb60d411a5c5b72b2e7d3527cfc84fd0-Abstract.html. (page 5, 7, 27, 30)

Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In Yennun Huang, Irwin King, Tie-Yan Liu, and Maarten van Steen (eds.), *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, pp. 2704–2710. ACM / IW3C2, 2020b. doi: 10.1145/3366423.3380027. URL https://doi.org/10.1145/3366423.3380027. (page 3)

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis R. Bach and David M. Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 448–456. JMLR.org, 2015. URL http://proceedings.mlr.press/v37/ioffe15.html. (page 28, 30)

Dongkwan Kim and Alice Oh. How to find your friendly neighborhood: Graph attention design with self-supervision. In *International Conference on Learning Representations (ICLR)*, 2021. (page 3)

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980. (page 26, 28, 30)

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SJU4ayYgl. (page 1, 2, 6, 7, 31, 32)

Boris Knyazev, Graham W. Taylor, and Mohamed R. Amer. Understanding attention and generalization in graph neural networks. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 4204–4214, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html. (page 1)

Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems (NeurIPS)*, 34, 2021. (page 3)

John Boaz Lee, Ryan A Rossi, Sungchul Kim, Nesreen K Ahmed, and Eunyee Koh. Attention models in graphs: A survey. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 13, 2019. (page 3)

Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence,*

EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019, pp. 4602–4609. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33014602. URL https://doi.org/10.1609/aaai.v33i01.33014602. (page 3)

P. Rigollet and J.-C. Hütter. High dimensional statistics. *Lecture notes for course 18S997*, 813:814, 2015. (page 21)

Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 2021. (page 6, 26)

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20, 2008. (page 1)

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3), 2008. (page 1)

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018. URL https://arxiv.org/abs/1811.05868. (page 6, 26, 27)

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. (page 28, 30)

Kiran K Thekumparampil, Chong Wang, Sewoong Oh, and Li-Jia Li. Attention-based graph neural network for semi-supervised learning. *arXiv preprint arXiv:1803.03735*, 2018. URL https://arxiv.org/abs/1803.03735. (page 3)

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html. (page 3)

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=rJXMpikCZ. (page 1, 2, 3, 6, 7, 31)

Guangtao Wang, Rex Ying, Jing Huang, and Jure Leskovec. Multi-hop attention graph neural networks. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2021a. (page 3)

Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*, 2020. (page 27)

Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019a. URL https://arxiv.org/abs/1909.01315. (page 29)

Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. Heterogeneous graph attention network. In Ling Liu, Ryen W. White, Amin Mantrach, Fabrizio Silvestri, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia (eds.), *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pp. 2022–2032. ACM, 2019b. doi: 10.1145/3308558.3313562. URL https://doi.org/10.1145/3308558.3313562. (page 3)

Yangkun Wang, Jiarui Jin, Weinan Zhang, Yong Yu, Zheng Zhang, and David Wipf. Bag of tricks for node classification with graph neural networks. *arXiv preprint arXiv:2103.13355*, 2021b. URL https://arxiv.org/abs/2103.13355. (page 32)

Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys (CSUR)*, 2020a. (page 1)

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32, 2020b. (page 3)

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km. (page 3, 32)

Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In Maria-Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 40–48. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/yanga16.html. (page 27)

Jiaxuan You, Zhitao Ying, and Jure Leskovec. Design space for graph neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 2020. (page 6, 27)

Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J. Kim. Graph transformer networks. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 11960–11970, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/9d63484abb477c97640154d40595a3bb-Abstract.html. (page 3)

Appendix

Table of Contents

A Theoretical results

A.1 A hard example for GCN

In this subsection, we present a dataset and classification task for which GCN performs poorly. Note that we follow the similar techniques and notation as (Fountoulakis et al., 2022), as described in the main paper.

We recall our data model. Fix $n, d \in \mathbb{N}$ and let $\varepsilon_1, \ldots, \varepsilon_n$ be i.i.d uniformly sampled from $\{-1, 0, 1\}$. Let $C_k = \{j \in [n] \mid \varepsilon_j = k\}$ for $k \in \{-1, 0, 1\}$. For each index $i \in [n]$, we set the feature vector $\mathbf{X}_i \in \mathbb{R}^d$ as $\mathbf{X}_i \sim \mathcal{N}(\varepsilon_i \cdot \boldsymbol{\mu}, \mathbf{I} \cdot \sigma^2)$, where $\boldsymbol{\mu} \in \mathbb{R}^d$, $\sigma \in \mathbb{R}$ and $\mathbf{I} \in \{0, 1\}^{d \times d}$ is the identity matrix. For a given pair $p, q \in [0, 1]$ we consider the stochastic adjacency matrix $\mathbf{A} \in \{0, 1\}^{n \times n}$ defined as follows. For $i, j \in [n]$ in the same class, we set $a_{ij} \sim \mathrm{Ber}(p)$, and if i, j are in different classes, we set $a_{ij} \sim \mathrm{Ber}(q)$. We let $\mathbf{D} \in \mathbb{R}^{n \times n}$ be a diagonal matrix containing the degrees of the vertices. We denote by $(\mathbf{X}, \mathbf{A}) \sim \mathsf{CSBM}(n, p, q, \boldsymbol{\mu}, \sigma^2)$ a sample obtained according to the above random process.

The task we wish to solve is classifying C_0 vs $C_{-1} \cup C_1$. Namely, we want our model φ to satisfy $\varphi(\mathbf{X}_i) < 0$ if and only if $i \in C_0$. Moreover, note that the posed problem *is not linearly classifiable*.

To this end, we start by stating an assumption on the choice of parameters. This assumption is necessary to achieve degree concentration in the graph.

Assumption 1. $p, q = \Omega(\log^2 n/n)$.

We now show the distribution of the convolved features. The following lemma can be easily obtained using the techniques in Baranwal et al. (2021).

Lemma 3. Fix p, q satisfying Assumption 1. With probability at least $1 - o(1)$ over $\mathbf{A}$ and $\{\varepsilon_i\}_i$,

$$(\mathbf{D}^{-1}\mathbf{A}\mathbf{X})_i \sim \mathcal{N}\left(\varepsilon_i \cdot \frac{p-q}{p+2q}\boldsymbol{\mu}, \frac{\sigma^2}{n(p+2q)}\right), \qquad \forall i \in [n].$$

To prove the above lemma, we need the following definition of our high probability event.

Definition 1. We define the even $\boldsymbol{\mathcal{E}}$ as the intersection of the following events over $\mathbf{A}$ and $\{\varepsilon_i\}_i$:

1. $\boldsymbol{\mathcal{E}}_1$ is the event that $|C_0| = \frac{n}{3} \pm O(\sqrt{n \log n})$, $|C_1| = \frac{n}{3} \pm O(\sqrt{n \log n})$ and $|C_{-1}| = \frac{n}{3} \pm O(\sqrt{n \log n})$.
2. $\boldsymbol{\mathcal{E}}_2$ is the event that for each $i \in [n]$, $\mathbf{D}_{ii} = \frac{n(p+2q)}{3}\left(1 \pm \frac{10}{\sqrt{\log n}}\right)$.
3. $\boldsymbol{\mathcal{E}}_3$ is the event that for each $i \in [n]$ and $k \in \{-1, 0, 1\}$,

$$|N_i \cap C_k| = \begin{cases} \mathbf{D}_{ii} \cdot \frac{p}{p+2q} \cdot \left(1 \pm \frac{10}{\sqrt{\log n}}\right) & \text{if } \ i \in C_k \\ \mathbf{D}_{ii} \cdot \frac{q}{p+2q} \cdot \left(1 \pm \frac{10}{\sqrt{\log n}}\right) & \text{if } \ i \notin C_k \end{cases}.$$

The following lemma is a direct application of Chernoff bound and a union bound.

Lemma 4. With probability at least $1 - 1/\mathrm{poly}(n)$ the event $\boldsymbol{\mathcal{E}}$ holds.

Proof of Lemma 3. By applying Lemma 4, and conditioned on $\boldsymbol{\mathcal{E}}$, for any $i \in [n]$

$$(\mathbf{D}^{-1}\mathbf{A}\mathbf{X})_i = \frac{1}{\mathbf{D}_{ii}} \sum_{j \in N_i} \mathbf{X}_j = \frac{1}{\mathbf{D}_{ii}} \left(\sum_{j \in N_i \cap C_{-1}} \mathbf{X}_j + \sum_{j \in N_i \cap C_0} \mathbf{X}_j + \sum_{j \in N_i \cap C_1} \mathbf{X}_j \right).$$

Using the definition of $\boldsymbol{\mathcal{E}}$ and properties of Gaussian distributions the lemma follows. □

Lemma 3 shows that essentially, the convolution reduced the variance and moved the means closer, but the structure of the problem stayed exactly the same. Therefore, one layer of GCN cannot separate C_0 from $C_{-1} \cup C_1$ with high probability.

A.2 A solution for GAT and CAT

In what follows, we show that GAT is able to handle the above classification task easily when the distance between the means is large enough. Then, we show how the additional convolution on the inputs to the score function improves the regime of perfect classification when the graph is not too noisy. Our main technical lemma considers a specific attention architecture and characterize the attention scores for our data model.

Lemma 5. Suppose that p, q satisfy Assumption 1, $\|\boldsymbol{\mu}\| \geq \omega\left(\sigma\sqrt{\log n}\right)$, fix the LeakyRelu constant $\beta \in (0,1)$ and $R \in \mathbb{R}$. Then, there exists a choice of attention architecture Ψ such that with probability at least $1 - o_n(1)$ over the data $(\mathbf{X}, \mathbf{A}) \sim \mathsf{CSBM}(n, p, q, \boldsymbol{\mu}, \sigma^2)$ the following holds.

$$\Psi(\mathbf{X}_i, \mathbf{X}_j) = \begin{cases} 10R\beta\|\boldsymbol{\mu}\|(1 \pm o(1)) & \text{if } i,j \in C_1^2 \\ -2R\|\boldsymbol{\mu}\|(1+2\beta)(1 \pm o(1)) & \text{if } i,j \in C_{-1}^2 \\ -2R\|\boldsymbol{\mu}\|(1+5\beta)(1 \pm o(1)) & \text{if } i \in C_1,\ j \in C_{-1} \\ 10R\beta\|\boldsymbol{\mu}\|(1 \pm o(1)) & \text{if } i \in C_{-1},\ j \in C_1 \\ -\frac{R}{2}\|\boldsymbol{\mu}\|(1-21\beta)(1 \pm o(1)) & \text{if } i \in C_0,\ j \in C_1 \\ -\frac{R}{2}\|\boldsymbol{\mu}\|(1-11\beta)(1 \pm o(1)) & \text{if } i \in C_0,\ j \in C_{-1} \\ -\frac{R}{2}\|\boldsymbol{\mu}\|(1-5\beta)(1 \pm o(1)) & \text{if } i \in C_1,\ j \in C_0 \\ -\frac{R}{2}\|\boldsymbol{\mu}\|(1-5\beta)(1 \pm o(1)) & \text{if } i \in C_{-1},\ j \in C_0 \\ 2R\beta\|\boldsymbol{\mu}\|(1 \pm o(1)) & \text{if } i,j \in C_0^2 \end{cases}.$$

Proof. We consider as an ansatz the following two layer architecture Ψ.

$$\tilde{\boldsymbol{w}} \stackrel{\text{def}}{=} \frac{\boldsymbol{\mu}}{\|\boldsymbol{\mu}\|}, \qquad \mathbf{S} \stackrel{\text{def}}{=} \begin{bmatrix} 1 & 1 \\ -1 & -1 \\ 1 & -1 \\ -1 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & -1 \\ -1 & 0 \end{bmatrix}, \qquad \boldsymbol{b} \stackrel{\text{def}}{=} \begin{bmatrix} -3/2 \\ -3/2 \\ -3/2 \\ -3/2 \\ -1/2 \\ -1/2 \\ -1/2 \\ -1/2 \end{bmatrix} \cdot \|\boldsymbol{\mu}\|, \qquad \boldsymbol{r} \stackrel{\text{def}}{=} R \cdot \begin{bmatrix} 2 \\ -2 \\ -2 \\ 2 \\ -1 \\ -1 \\ -1 \\ -1 \end{bmatrix},$$

where $R > 0$ is an arbitrary scaling parameter. The output of the attention model is defined as

$$\Psi(\mathbf{X}_i, \mathbf{X}_j) \stackrel{\text{def}}{=} \boldsymbol{r}^T \cdot \text{LeakyRelu}\left(\mathbf{S} \cdot \begin{bmatrix} \tilde{\boldsymbol{w}}^T\mathbf{X}_i \\ \tilde{\boldsymbol{w}}^T\mathbf{X}_j \end{bmatrix} + \boldsymbol{b}\right).$$

Let $\boldsymbol{\Delta}_{ij} \stackrel{\text{def}}{=} \mathbf{S} \cdot \begin{bmatrix} \tilde{\boldsymbol{w}}^T\mathbf{X}_i \\ \tilde{\boldsymbol{w}}^T\mathbf{X}_j \end{bmatrix} + \boldsymbol{b} \in \mathbb{R}^8$, and note that for each element $t \in [8]$ of $\boldsymbol{\Delta}_{ij}$, we have that $(\boldsymbol{\Delta}_{ij})_t = \mathbf{S}_{t,1}\tilde{\boldsymbol{w}}^T\mathbf{X}_i + \mathbf{S}_{t,2}\tilde{\boldsymbol{w}}^T\mathbf{X}_j + \boldsymbol{b}_t$. Note that the random variable $(\boldsymbol{\Delta}_{ij})_t$ is distributed as follows:

$$(\boldsymbol{\Delta}_{ij})_t \sim \begin{cases} \mathcal{N}\left((\mathbf{S}_{t,1}+\mathbf{S}_{t,2})\tilde{\boldsymbol{w}}^T\boldsymbol{\mu} + \boldsymbol{b}_t,\ \|\mathbf{S}_{t,*}\|^2\sigma^2\right) & \text{if } i,j \in C_1^2 \\ \mathcal{N}\left(-(\mathbf{S}_{t,1}+\mathbf{S}_{t,2})\tilde{\boldsymbol{w}}^T\boldsymbol{\mu} + \boldsymbol{b}_t,\ \|\mathbf{S}_{t,*}\|^2\sigma^2\right) & \text{if } i,j \in C_{-1}^2 \\ \mathcal{N}\left((\mathbf{S}_{t,1}-\mathbf{S}_{t,2})\tilde{\boldsymbol{w}}^T\boldsymbol{\mu} + \boldsymbol{b}_t,\ \|\mathbf{S}_{t,*}\|^2\sigma^2\right) & \text{if } i \in C_1,\ j \in C_{-1} \\ \mathcal{N}\left(-(\mathbf{S}_{t,1}-\mathbf{S}_{t,2})\tilde{\boldsymbol{w}}^T\boldsymbol{\mu} + \boldsymbol{b}_t,\ \|\mathbf{S}_{t,*}\|^2\sigma^2\right) & \text{if } i \in C_{-1},\ j \in C_1 \\ \mathcal{N}\left(\mathbf{S}_{t,2}\tilde{\boldsymbol{w}}^T\boldsymbol{\mu} + \boldsymbol{b}_t,\ \|\mathbf{S}_{t,*}\|^2\sigma^2\right) & \text{if } i \in C_0,\ j \in C_1 \\ \mathcal{N}\left(-\mathbf{S}_{t,2}\tilde{\boldsymbol{w}}^T\boldsymbol{\mu} + \boldsymbol{b}_t,\ \|\mathbf{S}_{t,*}\|^2\sigma^2\right) & \text{if } i \in C_0,\ j \in C_{-1} \\ \mathcal{N}\left(\mathbf{S}_{t,1}\tilde{\boldsymbol{w}}^T\boldsymbol{\mu} + \boldsymbol{b}_t,\ \|\mathbf{S}_{t,*}\|^2\sigma^2\right) & \text{if } i \in C_1,\ j \in C_0 \\ \mathcal{N}\left(-\mathbf{S}_{t,1}\tilde{\boldsymbol{w}}^T\boldsymbol{\mu} + \boldsymbol{b}_t,\ \|\mathbf{S}_{t,*}\|^2\sigma^2\right) & \text{if } i \in C_{-1},\ j \in C_0 \\ \mathcal{N}\left(\boldsymbol{b}_t,\ \|\mathbf{S}_{t,*}\|^2\sigma^2\right) & \text{if } i,j \in C_0^2 \end{cases}.$$

Therefore, for a fixed $i, j \in [n]^2$ we have that the entries of $\boldsymbol{\Delta}_{ij}$ are distributed as follows (where we use $\mathcal{N}_x^y$ as abbreviation for the Gaussian $\mathcal{N}(x, y)$)

$$\begin{aligned}
&\left[\mathcal{N}^{4\sigma^2}_{\frac{\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{4\sigma^2}_{\frac{-7\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{4\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{4\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{\frac{\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{\frac{\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}}\right] && \text{for } i,j \in C_1^2,\\
&\left[\mathcal{N}^{4\sigma^2}_{-\frac{7\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{4\sigma^2}_{\frac{\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{4\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{4\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{\frac{\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{\frac{\|\boldsymbol{\mu}\|}{2}}\right] && \text{for } i,j \in C_{-1}^2,\\
&\left[\mathcal{N}^{4\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{4\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{4\sigma^2}_{\frac{\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{4\sigma^2}_{-\frac{7\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{\frac{\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{\frac{\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}}\right] && \text{for } i,j \in C_1 \times C_{-1},\\
&\left[\mathcal{N}^{4\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{4\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{4\sigma^2}_{-\frac{7\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{4\sigma^2}_{\frac{\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{\frac{\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{\frac{\|\boldsymbol{\mu}\|}{2}}\right] && \text{for } i,j \in C_{-1} \times C_1,\\
&\left[\mathcal{N}^{4\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{4\sigma^2}_{-\frac{5\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{4\sigma^2}_{-\frac{5\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{4\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{\frac{\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}}\right] && \text{for } i,j \in C_0 \times C_1,\\
&\left[\mathcal{N}^{4\sigma^2}_{-\frac{5\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{4\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{4\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{4\sigma^2}_{-\frac{5\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{\frac{\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}}\right] && \text{for } i,j \in C_0 \times C_{-1},\\
&\left[\mathcal{N}^{4\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{4\sigma^2}_{-\frac{5\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{4\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{4\sigma^2}_{-\frac{5\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{\frac{\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}}\right] && \text{for } i,j \in C_1 \times C_0,\\
&\left[\mathcal{N}^{4\sigma^2}_{-\frac{5\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{4\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{4\sigma^2}_{-\frac{5\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{4\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{\frac{\|\boldsymbol{\mu}\|}{2}}\right] && \text{for } i,j \in C_{-1} \times C_0,\\
&\left[\mathcal{N}^{4\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{4\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{4\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{4\sigma^2}_{-\frac{3\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}} \quad \mathcal{N}^{\sigma^2}_{-\frac{\|\boldsymbol{\mu}\|}{2}}\right] && \text{for } i,j \in C_0^2,
\end{aligned}$$

Next, we will use the following lemma regarding LeakyRelu concentration.

Lemma 6 (Lemma A.6 in Fountoulakis et al. (2022))**.** Fix $s \in \mathbb{N}$, and let $z_1, \ldots, z_s$ be jointly Gaussian random variables with marginals $\boldsymbol{z}_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$. There exists an absolute constant $C > 0$ such that with probability at least $1 - o_s(1)$, we have

$$\text{LeakyRelu}(z_i) = \text{LeakyRelu}\left(\mu_i\right) \pm C\sigma_i\sqrt{\log s}, \quad \text{for all } i \in [s].$$

Using Lemma 6 with the assumption on $\|\boldsymbol{\mu}\|$ and a union bound, we have that with probability at least $1 - o_n(1)$, $\text{LeakyRelu}(\boldsymbol{\Delta}_{ij})$ is (up to $1 \pm o(1)$)

$$\begin{aligned}
&\left[\frac{\|\boldsymbol{\mu}\|}{2} \quad \frac{-7\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{3\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{3\beta\|\boldsymbol{\mu}\|}{2} \quad \frac{\|\boldsymbol{\mu}\|}{2} \quad \frac{\|\boldsymbol{\mu}\|}{2} \quad -\frac{3\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{3\beta\|\boldsymbol{\mu}\|}{2}\right] && \text{for } i,j \in C_1^2,\\
&\left[-\frac{7\beta\|\boldsymbol{\mu}\|}{2} \quad \frac{\|\boldsymbol{\mu}\|}{2} \quad -\frac{3\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{3\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{3\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{3\beta\|\boldsymbol{\mu}\|}{2} \quad \frac{\|\boldsymbol{\mu}\|}{2} \quad \frac{\|\boldsymbol{\mu}\|}{2}\right] && \text{for } i,j \in C_{-1}^2,\\
&\left[-\frac{3\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{3\beta\|\boldsymbol{\mu}\|}{2} \quad \frac{\|\boldsymbol{\mu}\|}{2} \quad -\frac{7\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{3\beta\|\boldsymbol{\mu}\|}{2} \quad \frac{\|\boldsymbol{\mu}\|}{2} \quad \frac{\|\boldsymbol{\mu}\|}{2} \quad -\frac{3\beta\|\boldsymbol{\mu}\|}{2}\right] && \text{for } i,j \in C_1 \times C_{-1},\\
&\left[-\frac{3\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{3\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{7\beta\|\boldsymbol{\mu}\|}{2} \quad \frac{\|\boldsymbol{\mu}\|}{2} \quad \frac{\|\boldsymbol{\mu}\|}{2} \quad -\frac{3\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{3\beta\|\boldsymbol{\mu}\|}{2} \quad \frac{\|\boldsymbol{\mu}\|}{2}\right] && \text{for } i,j \in C_{-1} \times C_1,\\
&\left[-\frac{\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{5\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{5\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{\beta\|\boldsymbol{\mu}\|}{2} \quad \frac{\|\boldsymbol{\mu}\|}{2} \quad -\frac{\|\boldsymbol{\mu}\|}{2} \quad -\frac{3\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{\|\beta\boldsymbol{\mu}\|}{2}\right] && \text{for } i,j \in C_0 \times C_1,\\
&\left[-\frac{5\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{5\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{3\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{\beta\|\boldsymbol{\mu}\|}{2} \quad \frac{\|\boldsymbol{\mu}\|}{2} \quad -\frac{\beta\|\boldsymbol{\mu}\|}{2}\right] && \text{for } i,j \in C_0 \times C_{-1},\\
&\left[-\frac{\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{5\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{5\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{\beta\|\boldsymbol{\mu}\|}{2} \quad \frac{\|\boldsymbol{\mu}\|}{2} \quad -\frac{\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{3\beta\|\boldsymbol{\mu}\|}{2}\right] && \text{for } i,j \in C_1 \times C_0,\\
&\left[-\frac{5\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{5\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{3\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{\|\beta\boldsymbol{\mu}\|}{2} \quad \frac{\|\boldsymbol{\mu}\|}{2}\right] && \text{for } i,j \in C_{-1} \times C_0,\\
&\left[-\frac{3\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{3\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{3\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{3\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{\beta\|\boldsymbol{\mu}\|}{2} \quad -\frac{\beta\|\boldsymbol{\mu}\|}{2}\right] && \text{for } i,j \in C_0^2.
\end{aligned}$$

Then,

$$\boldsymbol{r}^T \cdot \text{LeakyRelu}(\boldsymbol{\Delta}_{ij}) = \begin{cases} 10R\beta\|\boldsymbol{\mu}\|(1 \pm o(1)) & \text{if } i,j \in C_1^2 \\ -2R\|\boldsymbol{\mu}\|(1+2\beta)(1 \pm o(1)) & \text{if } i,j \in C_{-1}^2 \\ -2R\|\boldsymbol{\mu}\|(1+5\beta)(1 \pm o(1)) & \text{if } i \in C_1,\ j \in C_{-1} \\ 10R\beta\|\boldsymbol{\mu}\|(1 \pm o(1)) & \text{if } i \in C_{-1},\ j \in C_1 \\ -\frac{R}{2}\|\boldsymbol{\mu}\|(1-21\beta)(1 \pm o(1)) & \text{if } i \in C_0,\ j \in C_1 \\ -\frac{R}{2}\|\boldsymbol{\mu}\|(1-11\beta)(1 \pm o(1)) & \text{if } i \in C_0,\ j \in C_{-1} \\ -\frac{R}{2}\|\boldsymbol{\mu}\|(1-5\beta)(1 \pm o(1)) & \text{if } i \in C_1,\ j \in C_0 \\ -\frac{R}{2}\|\boldsymbol{\mu}\|(1-5\beta)(1 \pm o(1)) & \text{if } i \in C_{-1},\ j \in C_0 \\ 2R\beta\|\boldsymbol{\mu}\|(1 \pm o(1)) & \text{if } i,j \in C_0^2 \end{cases},$$

and the proof is complete. □

Next we will define our high probability event.

Definition 2. $\boldsymbol{\mathcal{E}}' \stackrel{\text{def}}{=} \boldsymbol{\mathcal{E}} \cap \boldsymbol{\mathcal{E}}^*$, where $\boldsymbol{\mathcal{E}}^*$ is the event that for a fixed $\boldsymbol{w} \in \mathbb{R}^d$, all $i \in [n]$ satisfy $|\boldsymbol{w}^T\mathbf{X}_i - \mathbf{E}[\boldsymbol{w}^T\mathbf{X}_i]| \le 10\sigma\|\boldsymbol{w}\|_2\sqrt{\log n}$.

The following lemma is obtained by using Lemma 4 with standard Gaussian concentration and a union bound.

Lemma 7. With probability at least $1 - 1/\text{poly}(n)$ event $\boldsymbol{\mathcal{E}}'$ holds.

Corollary 8. Suppose that p, q satisfy Assumption 1, $\|\boldsymbol{\mu}\| = \omega(\sigma\sqrt{\log n})$ and fix $R \in \mathbb{R}$. Then, there exists a choice of attention architecture Ψ such that with probability $1 - o_n(1)$ over $(\mathbf{A}, \mathbf{X}) \sim \mathsf{CSBM}(n, p, q, \boldsymbol{\mu}, \sigma^2)$ it holds that

$$\gamma_{ij} = \begin{cases} \frac{3}{np}(1 \pm o(1)) & \text{if } i,j \in C_0^2 \cup C_1^2 \\ \frac{3}{nq}(1 \pm o(1)) & \text{if } i,j \in C_{-1} \times C_1 \\ \frac{3}{nq}\exp(-\Theta(R\|\boldsymbol{\mu}\|)) & \text{if } i,j \in C_{-1} \times C_{-1} \cup C_0 \\ \frac{3}{np}\exp(-\Theta(R\|\boldsymbol{\mu}\|)) & \text{otherwise} \end{cases},$$

where R is a parameter of the architecture.

Proof. The proof is immediate. First applying the ansatz from Lemma 5 with $\beta < 1/25$, Lemma 7 and a union bound. Using the definition of γ_{ij} concludes the proof. □

Next, we prove Thm. 1 that the model distinguish nodes from C_0 for any choice of p, q satisfying Assumption 1. We restate the theorem for convince.

Theorem 9 (Formal restatement of Thm. 1)**.** Suppose that p, q satisfy Assumption 1 and $\|\boldsymbol{\mu}\|_2 = \omega(\sigma\sqrt{\log n})$. Then, there exists a choice of attention architecture Ψ such that with probability at least $1 - o_n(1)$ over the data $(\mathbf{X}, \mathbf{A}) \sim \mathsf{CSBM}(n, p, q, \boldsymbol{\mu}, \sigma^2)$, the estimator

$$\hat{x}_i \stackrel{\text{def}}{=} \sum_{j \in N_i} \gamma_{ij}\tilde{\boldsymbol{w}}^T\mathbf{X}_j + b \ \text{ where } \tilde{\boldsymbol{w}} = \boldsymbol{\mu}/\|\boldsymbol{\mu}\|,\ b = -\|\boldsymbol{\mu}\|/2$$

satisfies $\hat{x}_i < 0$ if and only if $i \in C_0$.

Proof. Let Ψ be the architecture from Cor. 8 and let R satisfy $R\|\boldsymbol{\mu}\|_2 = \omega(1)$. We will compute the mean and variance of the estimator $\hat{x}_i$ conditioned on $\boldsymbol{\mathcal{E}}'$. Suppose that $i \in C_0$. By using Cor. 8, Definition 2 and our assumption on $\|\boldsymbol{\mu}\|$ and R, we have

$$\max\left\{\frac{3}{np}\exp(-\Theta(R\|\boldsymbol{\mu}\|)), \frac{3}{nq}\exp(-\Theta(R\|\boldsymbol{\mu}\|))\right\} = o\left(\frac{1}{n(p+2q)}\right),$$

and therefore

$$\begin{aligned}
\mathbf{E}\left[\hat{x}_i \mid \boldsymbol{\mathcal{E}}'\right] &= \mathbf{E}\left[\sum_{k\in\{-1,0,1\}}\sum_{j\in N_i\cap C_k}\gamma_{ij}\tilde{\boldsymbol{w}}^T\mathbf{X}_j \;\middle|\; \boldsymbol{\mathcal{E}}'\right] - \frac{\|\boldsymbol{\mu}\|}{2} \\
&= \mathbf{E}[|C_0\cap N_i| \mid \boldsymbol{\mathcal{E}}']\left(\pm\frac{3}{np}(1\pm o(1))\cdot 10\sigma\sqrt{\log n}\right) \\
&\quad + \mathbf{E}[|C_1\cap N_i| \mid \boldsymbol{\mathcal{E}}']\left(o\left(\frac{1}{n(p+2q)}\right)\cdot(\|\boldsymbol{\mu}\|\pm 10\sigma\sqrt{\log n})\right) \\
&\quad + \mathbf{E}[|C_{-1}\cap N_i| \mid \boldsymbol{\mathcal{E}}']\left(o\left(\frac{1}{n(p+2q)}\right)\cdot(-\|\boldsymbol{\mu}\|\pm 10\sigma\sqrt{\log n})\right) - \frac{\|\boldsymbol{\mu}\|}{2} \\
&= -\frac{\|\boldsymbol{\mu}\|}{2}(1\pm o(1)).
\end{aligned}$$

By similar reasoning we have that for $i\in C_{-1}\cup C_1$, $\mathbf{E}\left[\hat{x}_i \mid \boldsymbol{\mathcal{E}}'\right] = \frac{\|\boldsymbol{\mu}\|}{2}(1\pm o(1))$.

Next, we claim that for each $i\in[n]$ the random variable $\hat{x}_i$ given $\boldsymbol{\mathcal{E}}'$ is sub-Gaussian with a small sub-Gaussian constant compared to the above expectation. The following lemma is a straightforward adaptation of Lemma A.11 in Fountoulakis et al. (2022), and we provide its proof for completeness.

Lemma 10. Conditioned on $\boldsymbol{\mathcal{E}}'$, the random variables $\{\hat{\boldsymbol{x}}_i\}_i$ are sub-Gaussian with parameter $\tilde{\sigma}_i^2 = O\left(\frac{\sigma^2}{np}\right)$ if $i\in C_0\cup C_1$ and $\tilde{\sigma}_i^2 = O\left(\frac{\sigma^2}{nq}\right)$ otherwise.

Proof. Fix $i\in[n]$, and write $\mathbf{X}_i = \varepsilon_i\boldsymbol{\mu}+\sigma\boldsymbol{g}_i$ where $\boldsymbol{g}_i\sim\mathcal{N}(0,\mathbf{I}_d)$, and ε_i denotes the class membership. Consider $\hat{x}_i$ as a function of $\boldsymbol{g} = [\boldsymbol{g}_1\circ\boldsymbol{g}_2\circ\cdots\circ\boldsymbol{g}_n]\in\mathbb{R}^{nd}$, where $\circ$ denotes vertical concatenation. Namely, consider the function

$$\hat{x}_i = f_i(\boldsymbol{g}) \stackrel{\text{def}}{=} \sum_{j\in N_i}\gamma_{ij}(\boldsymbol{g})\,\tilde{\boldsymbol{w}}^T(\varepsilon_j\boldsymbol{\mu}+\sigma\boldsymbol{g}_j) - \|\boldsymbol{\mu}\|/2, \quad i\in[n].$$

Since $\boldsymbol{g}\sim\mathcal{N}(0,\mathbf{I}_{nd})$, proving that $\hat{x}_i$ given $\boldsymbol{\mathcal{E}}'$ is sub-Gaussian for each $i\in[n]$, reduces to showing that the function $f_i:\mathbb{R}^{nd}\to\mathbb{R}$ is Lipschitz over $E\subseteq\mathbb{R}^{nd}$ defined by $\boldsymbol{\mathcal{E}}'$ and the relation $\mathbf{X}_i = \varepsilon_i\boldsymbol{\mu}+\sigma\boldsymbol{g}_i$. That is, $E\stackrel{\text{def}}{=}\left\{\boldsymbol{g}\in\mathbb{R}^{nd}\;\middle|\;|\tilde{\boldsymbol{w}}^T\boldsymbol{g}_i|\le 10\sqrt{\log n},\forall i\in[n]\right\}$. Specifically, we show that conditioning on the event $\boldsymbol{\mathcal{E}}'$ (which restricts $\boldsymbol{g}\in E$), the Lipschitz constant L_{f_i} of f_i satisfies $L_{f_i} = O\left(\frac{\sigma}{\sqrt{np}}\right)$ for $i\in C_0\cup C_1$ and $L_{f_i} = O\left(\frac{\sigma}{nq}\right)$ otherwise, and hence proving the claim.

First note that event $\boldsymbol{\mathcal{E}}'$ induce a transformation which transforms isotropic Gaussians to truncated Gaussians vectors. Similarly to Fountoulakis et al. (2022), we can show that this transformation can be obtained by a push-forward mapping whose Lipschitz constant is 1.

$$\bar{\boldsymbol{v}} = M(\boldsymbol{v}) \stackrel{\text{def}}{=} [\tau(\boldsymbol{v}_1),\tau(\boldsymbol{v}_2),\ldots,\tau(\boldsymbol{v}_n)]^T \tag{8}$$

where $\tau(x)\stackrel{\text{def}}{=}\Phi^{-1}((1-2c)\Phi(x)+c)$ for $c=\Phi(-10\sqrt{\log n})$.

To compute the Lipschitz constant of $f_i(\boldsymbol{g})$ for $i\in[n]$, let us denote $\mathbf{X} = [\mathbf{X}_1\circ\mathbf{X}_2\circ\cdots\circ\mathbf{X}_n]$ and consider the function

$$\tilde{f}_i(\mathbf{X}) \stackrel{\text{def}}{=} \sum_{j\in N_i}\gamma_{ij}(\mathbf{X})\,\tilde{\boldsymbol{w}}^T\mathbf{X}_j, \quad i\in[n]$$

Let us assume without loss of generality that $i\in C_0$ (the cases for $i\in C_1$ and $i\in C_{-1}$ are obtained identically). Conditioning on the event $\boldsymbol{\mathcal{E}}'$, which imposes the restriction that $\mathbf{X}\in\tilde{E}$ where

$$\tilde{E}\stackrel{\text{def}}{=}\left\{\mathbf{X}\in\mathbb{R}^{nd}\;\middle|\;|\mathbf{X}_i-\varepsilon_i\boldsymbol{\mu}|\le 10\sigma\sqrt{\log n},\forall i\in[n]\right\}.$$

Conditioning on $\boldsymbol{\mathcal{E}}'$ (which restricts $\mathbf{X}, \mathbf{X}' \in \tilde{E}$), using Cor. 8 and recalling that R satisfies $R\|\boldsymbol{\mu}\|_2 = \omega(1)$, we get[3]

$$
\begin{aligned}
&\left|\tilde{f}_i(\mathbf{X}) - \tilde{f}_i(\mathbf{X}')\right| \\
&\simeq \left| \sum_{j \in N_i \cap C_0} \frac{3}{np} \tilde{\boldsymbol{w}}^T (\mathbf{X}_j - \mathbf{X}'_j) + \sum_{j \in N_i \cap C_1} \frac{3}{np} \cdot e^{-\Theta(R\|\boldsymbol{\mu}\|_2)} \tilde{\boldsymbol{w}}^T (\mathbf{X}_j - \mathbf{X}'_j) + \sum_{j \in N_i \cap C_{-1}} \frac{3}{np} \cdot e^{-\Theta(R\|\boldsymbol{\mu}\|_2)} \tilde{\boldsymbol{w}}^T (\mathbf{X}_j - \mathbf{X}'_j) \right| \\
&= \left| \begin{bmatrix} \frac{3}{np}(1 \pm o(1))\tilde{\boldsymbol{w}} & \text{if } j \in N_i \cap C_0 \\ \frac{3}{np} \exp(-\Theta(R\|\boldsymbol{\mu}\|_2))(1 \pm o(1))\tilde{\boldsymbol{w}} & \text{if } j \in N_i \cap C_1 \\ \frac{3}{np} \exp(-\Theta(R\|\boldsymbol{\mu}\|_2))(1 \pm o(1))\tilde{\boldsymbol{w}} & \text{if } j \in N_i \cap C_{-1} \\ 0 & \text{if } j \notin N_i \end{bmatrix}^T_{j \in [n]} (\mathbf{X} - \mathbf{X}') \right| \\
&\leq \left\| \begin{bmatrix} \frac{3}{np}(1 \pm o(1))\tilde{\boldsymbol{w}} & \text{if } j \in N_i \cap C_0 \\ \frac{3}{np} \exp(-\Theta(R\|\boldsymbol{\mu}\|_2))(1 \pm o(1))\tilde{\boldsymbol{w}} & \text{if } j \in N_i \cap C_1 \\ \frac{3}{np} \exp(-\Theta(R\|\boldsymbol{\mu}\|_2))(1 \pm o(1))\tilde{\boldsymbol{w}} & \text{if } j \in N_i \cap C_{-1} \\ 0 & \text{if } j \notin N_i \end{bmatrix}_{j \in [n]} \right\|_2 \|\mathbf{X} - \mathbf{X}'\|_2 \\
&\leq \sqrt{\frac{3}{np}}(1 + o(1))\|\tilde{\boldsymbol{w}}\|_2 \|\mathbf{X} - \mathbf{X}'\|_2 \\
&= \sqrt{\frac{3}{np}}(1 + o(1)) \|\mathbf{X} - \mathbf{X}'\|_2 .
\end{aligned}
$$

This shows the Lipschitz constant of $\tilde{f}_i(\mathbf{X})$ over $\tilde{E}$ satisfies $L_{\tilde{f}_i} = O\left(\frac{1}{\sqrt{np}}\right)$. On the other hand, by viewing $\mathbf{X}$ as a function of $\boldsymbol{g}$, it is straightforward to see that the function $h(\boldsymbol{g}) : \mathbb{R}^{nd} \to \mathbb{R}^{nd}$ defined by $h(\boldsymbol{g}) \stackrel{\text{def}}{=} \mathbf{X}(\boldsymbol{g})$ has Lipschitz constant $L_h = \sigma$, as

$$\|h(\boldsymbol{g}) - h(\boldsymbol{g}')\|_2 = \|\boldsymbol{\varepsilon}\boldsymbol{\mu} + \sigma\boldsymbol{g} - (\boldsymbol{\varepsilon}\boldsymbol{\mu} + \sigma\boldsymbol{g}')\|_2 = \sigma\|\boldsymbol{g} - \boldsymbol{g}'\|_2.$$

Therefore, since $f_i(\boldsymbol{g}) = \tilde{f}_i(h(\boldsymbol{g}))$ and $\boldsymbol{g} \in E$ if and only if $\mathbf{X} \in \tilde{E}$, we have that, conditioning on $\boldsymbol{\mathcal{E}}'$, the function $\hat{x}_i = f_i(\boldsymbol{g})$ is Lipschitz continuous with Lipschitz constant $L_{f_i} = L_{\tilde{f}_i} L_h = O\left(\frac{\sigma}{\sqrt{np}}\right)$. Since $\boldsymbol{g} \sim \mathcal{N}(0, \mathbf{I}_{nd})$, we know that $\hat{x}_i$ is sub-Gaussian with sub-Gaussian constant $\tilde{\sigma}^2 = L^2_{f_i} = O\left(\frac{\sigma^2}{np}\right)$. □

The following lemma will be used for bounding the misclassification probability.

Lemma 11 (Rigollet & Hütter (2015))**.** Let $x_1, \ldots, x_n$ be sub-Gaussian random variables with the same mean and sub-Gaussian parameter $\tilde{\sigma}^2$. Then,

$$\mathbf{E}\left[\max_{i \in [n]} (x_i - \mathbf{E}[x_i])\right] \leq \tilde{\sigma}\sqrt{2 \log n}.$$

Moreover, for any $t > 0$

$$\mathbf{Pr}\left[\max_{i \in [n]} (x_i - \mathbf{E}[x_i]) > t\right] \leq 2n \exp\left(-\frac{t^2}{2\tilde{\sigma}^2}\right).$$

We bound the probability of misclassification

$$\mathbf{Pr}\left[\max_{i \in C_0} \hat{x}_i \geq 0\right] \leq \mathbf{Pr}\left[\max_{i \in C_0} \hat{x}_i > t + \mathbf{E}[\hat{x}_i]\right],$$

for $t < |\mathbf{E}[\hat{x}_i]| = \frac{\|\boldsymbol{\mu}\|_2}{2}(1 \pm o(1))$. By Lemma 10, picking $t = \Theta\left(\sigma\sqrt{\log |C_0|}\right)$ and applying Lemma 11 implies that the above probability is $1/\text{poly}(n)$.

[3] We drop the $(1 \pm o(1))$ in the first line of the computation for compactness and use $\simeq$ as notation.

Similarly for class $C_1 \cup C_{-1}$ we have that the misclassification probability is

$$\begin{aligned}\mathbf{Pr}\left[\min_{i\in C_1\cup C_{-1}} \hat{x}_i \le 0\right] &= \mathbf{Pr}\left[-\max_{i\in C_1\cup C_{-1}} (-\hat{x}_i) \le 0\right] = \mathbf{Pr}\left[\max_{i\in C_1\cup C_{-1}} (-\hat{x}_i) \ge 0\right] \\ &\le \mathbf{Pr}\left[\max_{i\in C_1\cup C_{-1}} -\hat{x}_i > t - \mathbf{E}[\hat{x}_i]\right],\end{aligned}$$

for $t < \mathbf{E}[\hat{x}_i]$. Picking $t = \Theta\left(\sigma\sqrt{\log|C_1 \cup C_{-1}|}\right)$ and applying Lemma 11 and a union bound over the misclassification probabilities of both classes conclude the proof of the corollary. □

Combining Thm. 9 with Lemma 3, we immediately get Cor. 2 which we restate below.

Corollary 12. Suppose $p, q = \Omega(\log^2 n/n)$ and $\|\boldsymbol{\mu}\| \ge \omega\left(\sigma\sqrt{\frac{(p+2q)\log n}{n(p-q)^2}}\right)$. Then, there is a choice of attention architecture Ψ such that, with probability at least $1 - o(1)$ over the data $(\mathbf{X}, \mathbf{A}) \sim \mathsf{CSBM}(n, p, q, \boldsymbol{\mu}, \sigma^2)$, CAT separates nodes C_0 from $C_1 \cup C_{-1}$.

B Synthetic experiments

In this section, we present the complete results for the synthetic data experiments of §6.1. First, we describe the parameterization we use for the 1-layer GCN, GAT, and CAT models; then, we verify the behavior of the normalized score function (γ_{ij}) matches that of the theory presented in Cor. 8. In particular, we visualize the average of the following three groups of gammas (Fig. 4):

- Gammas γ_{ij} included in $i, j \in C_0^2 \cup C_1^2$. Solid lines.
- Gammas γ_{ij} included in $i, j \in C_{-1} \times C_1$. Dashed lines.
- The rest of gammas. Dotted lines.

For completeness, we also include the empirical results that validate Thm. 1 and Cor. 2, which were discussed already in §6.1.

Experimental setup. We assume the following parametrization for the 1-layer GCN, GAT, and CAT:

$$\boldsymbol{h}'_i = \left(\sum_{j \in N_i} \gamma_{ij} \tilde{\boldsymbol{w}}^T \mathbf{X}_j \right) - C \cdot \|\boldsymbol{\mu}\|/2 \,, \tag{9}$$

where N_i are the set of neighbors of node i, $\mathbf{X}_j$ are the features of node j—obtained from the CSBM described in §4, and $\boldsymbol{h}'_i$ are the logits of the prediction of node i. Note that for GCN we have $\gamma_{ij} = \frac{1}{|N_i^*|}$. Otherwise, we consider the following parameterization of the score function Ψ:

$$\gamma_{ij} = \frac{\exp\left(\Psi(\boldsymbol{h}_i, \boldsymbol{h}_j)\right)}{\sum_{k \in N_i^*} \exp(\Psi(\boldsymbol{h}_i, \boldsymbol{h}_k))} \quad \text{where} \tag{10}$$

$$\Psi(\boldsymbol{h}_i, \boldsymbol{h}_j) \stackrel{\text{def}}{=} \boldsymbol{r}^T \cdot \text{LeakyRelu}\left(\mathbf{S} \cdot \begin{bmatrix} \tilde{\boldsymbol{w}}^T \boldsymbol{h}_i \\ \tilde{\boldsymbol{w}}^T \boldsymbol{h}_j \end{bmatrix} + \boldsymbol{b} \right) . \tag{11}$$

For these experiments, we define the parameters $\tilde{\boldsymbol{w}}$, $\mathbf{S}$, $\boldsymbol{b}$ and $\boldsymbol{r}$ as in the proofs in App. A:

$$\tilde{\boldsymbol{w}} \stackrel{\text{def}}{=} \frac{\boldsymbol{\mu}}{\|\boldsymbol{\mu}\|}, \qquad \mathbf{S} \stackrel{\text{def}}{=} \begin{bmatrix} 1 & 1 \\ -1 & -1 \\ 1 & -1 \\ -1 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & -1 \\ -1 & 0 \end{bmatrix}, \qquad \boldsymbol{b} \stackrel{\text{def}}{=} \begin{bmatrix} -3/2 \\ -3/2 \\ -3/2 \\ -3/2 \\ -1/2 \\ -1/2 \\ -1/2 \\ -1/2 \end{bmatrix} \cdot \|\boldsymbol{\mu}\| \cdot C, \qquad \boldsymbol{r} \stackrel{\text{def}}{=} R \cdot \begin{bmatrix} 2 \\ -2 \\ -2 \\ 2 \\ -1 \\ -1 \\ -1 \\ -1 \end{bmatrix}, \tag{12}$$

where $R > 0$ and $C > 0$ are arbitrary scaling parameters. Both C and R and input to the score function are set different for each of the models, as indicated in Table 4. In particular, we set $R = \frac{7}{\|\boldsymbol{\mu}\|}$ for both GAT and CAT such that: i) all γ_{ij} are distinguishable as we decrease $\|\boldsymbol{\mu}\|$; and ii) we avoid numerical instabilities in the implementation when computing the exponential of $R \times \|\boldsymbol{\mu}\|$ in order to obtain γ_{ij} (see Cor. 8), as the exponential of small or large values leads to under/overflow issues. As for C, we set $C = 1$ for GAT and $C = (p-q)/(p+2q)$ for CAT such that we counteract the fact that the distance between classes shrink as we increase q, see Lemma 3:

Regarding the data model, we set (as described in §6.1) $n = 10000$, $p = 0.5$, $\sigma = 0.1$, and $d = n/\left(5\log^2(n)\right)$. We set the slope of the LeakyReLU activation to $\beta = 1/5$ for the GAT and $\beta = 0.01$ for CAT, such that the proof of Cor. 8 is valid. As described in the main paper, to assess the sensitivity to structural noise, we present the complete results for two sets of experiments. First, we vary the noise level q between 0 and 0.5, fixing the mean vector $\boldsymbol{\mu}$. We test two values of $\|\boldsymbol{\mu}\|$: the first corresponds to the *easy* regime ($\|\boldsymbol{\mu}\| = 10\sigma\sqrt{2\log n} \approx 4.3$) where classes are far apart; and the second correspond to the *hard* regime ($\|\boldsymbol{\mu}\| = \sigma = 0.1$)

Table 4: Parameters for the synthetic experiments.

Model	C	R	$\boldsymbol{h}_i$
GCN	0	–	–
GAT	1	$\frac{7}{\|\boldsymbol{\mu}\|}$	$\mathbf{X}_i$
CAT	$\frac{p-q}{p+2q}$	$\frac{7}{\|\boldsymbol{\mu}\|}$	$\frac{1}{\|N_i^*\|}\sum_{k\in N_i^*}\mathbf{X}_k$

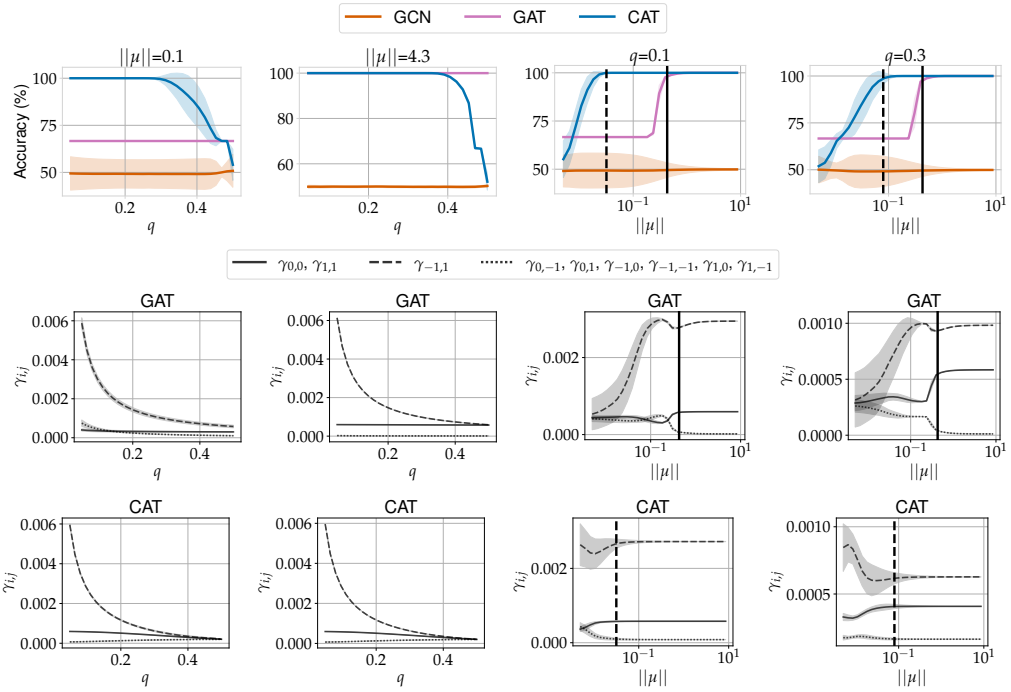


Figure 4: Synthetic data results. On the top row, we show the node classification, and in the following two rows we show the γ_{ij} values for GAT and CAT respectively. In the two left-most figures, we show how the results vary with the noise level q for $\|\boldsymbol{\mu}\| = 0.1$ and $\|\boldsymbol{\mu}\| = 4.3$. In the two right-most figures, we show how the results vary with the norm of the means $\|\boldsymbol{\mu}\|$ for $q = 0.1$ and $q = 0.3$. We use two vertical lines to present the classification threshold stated in Thm. 1 (solid line) and Cor. 2 (dashed line).

where the distance between the clusters is small. In the second experiment we modify instead the distance between the means, sweeping $\|\boldsymbol{\mu}\|$ in the range $\left[\sigma/20, 20\sigma\sqrt{2\log n}\right]$ which corresponds to $[0.005, 8.58]$, and thus covering the transition from the hard setting (small $\|\boldsymbol{\mu}\|$) to the easy one (large $\|\boldsymbol{\mu}\|$). In these experiments, we fix q to 0.1 (low noise) and 0.3 (high noise).

Results are summarized in Fig. 4. The top row contains the node classification performance for each of the models (i.e., Fig. 1), the next two rows contain the γ_{ij} values for GAT and CAT respectively. The two left-most columns of Fig. 4 show the results for the hard and easy regimes, respectively, as we vary the noise level q. In the hard regime, we observe that GAT is unable to achieve separation for any value of q, whereas CAT achieves perfect classification when q is small enough. The gamma plots help shed some light on this question. For GAT, we observe that the gammas represented with the dotted and solid lines collapse for any value of q (see middle plot), while this does not happen for CAT when the noise level is low (see bottom plot). This exemplifies the advantage of CAT over GAT as stated in Cor. 2. When the distance between the means is large enough, we see that GAT achieves perfect results independently of q, as stated in Thm. 1. We also observe that, in this case,

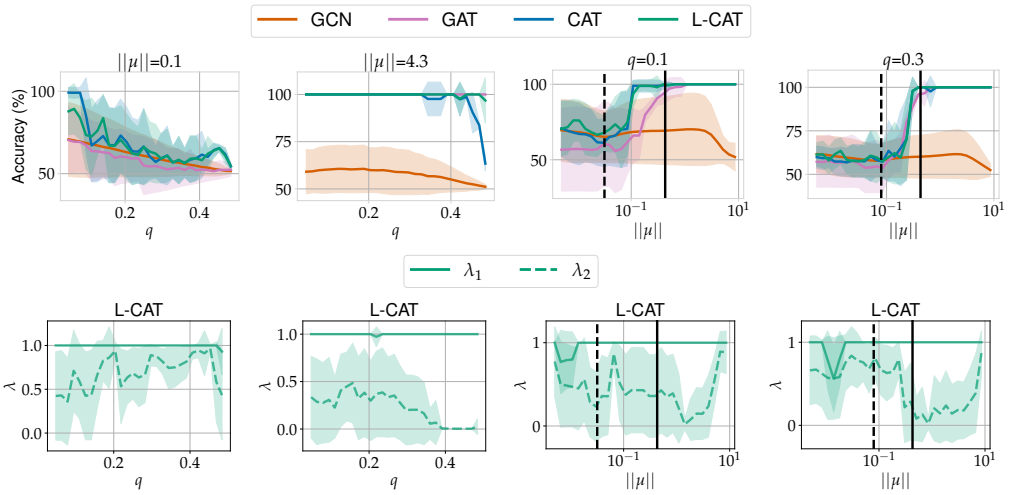


Figure 5: Synthetic data results learning C, λ_1 and λ_2. On the top row, we show the node classification accuracy, and in the bottom row we show the learned values of λ_1 and λ_2 for L-CAT. In the two left-most figures, we show how the results vary with the noise level q for $\|\boldsymbol{\mu}\| = 0.1$ and $\|\boldsymbol{\mu}\| = 4.3$. In the two right-most figures, we show how the results vary with the norm of the means $\|\boldsymbol{\mu}\|$ for $q = 0.1$ and $q = 0.3$. We use two vertical lines to present the classification threshold stated in Thm. 1 (solid line) and Cor. 2 (dashed line).

the gammas represented with the dotted and solid lines do not collapse for any value of q. In contrast, as we increase q, CAT fails to satisfy the condition in Cor. 2, and therefore achieves inferior performance. We note that the low performance is due to the fact that all gammas collapse to the same value for large noise levels.

For the second set of experiments (two right-most columns of Fig. 1), where we fix q and sweep $\|\boldsymbol{\mu}\|$, we observe that, for both values of q, there exists a transition in the accuracy of both GAT and CAT as a function of $\|\boldsymbol{\mu}\|$. As shown in the main manuscript, GAT achieves perfect accuracy when the distance between the means satisfies the condition in Thm. 1 (solid vertical line in Fig. 1). Moreover, we can see the improvement CAT obtains over GAT. Indeed, when $\|\boldsymbol{\mu}\|$ satisfies the conditions of Cor. 2 (dashed vertical line in Fig. 1), the classification threshold is improved. As we increase q, we see that the gap between the two vertical lines decreases, which means that the improvement decreases as q increments, exactly as stated in Cor. 2. This transition from the hard regime to the easy regime is also observed in the gamma plots: we observe the largest difference in value between the different groups of lambdas for values of $\|\boldsymbol{\mu}\|$ that satisfy the condition in Thm. 1 (that is to the right of the vertical lines).

B.1 Other experiments

In the following, we extend the results for the synthetic data presented above. In particular, we aim to evaluate if L-CAT is able to achieve top performance regardless of the scenario. That is, we want to evaluate if L-CAT consistently performs at least as good as the best-performing model. We change the fixed-parameter setting of the previous section and, instead, we evaluate the performance of GCN, GAT, CAT and L-CAT when we learn the model-dependent parameters.

Experimental setup. We assume the same parametrization for the 1-layer GCN, GAT and CAT described in Eq. 9 and Eq. 11. For L-CAT, we add the parameters λ_1 and λ_2, as indicated in Eq. 7. We fix the parameters shared among the models, that is, $\tilde{\boldsymbol{w}}$, $\mathbf{S}$, $\boldsymbol{b}$, $\boldsymbol{r}$, and R, with the values indicated in Eq. 12. Different from previous experiments, we now learn C and, for L-CAT, we also learn λ_1 and λ_2. We choose to fix part of the parameters (instead of learning them all) to keep the problem as similar as possible to the theoretical analysis we provided in §4 and App. A. If we instead learn all the parameters, it takes a single dimension

of the features to be (close to) linearly separable to find a solution that achieves a similar performance regardless of the model, which hinders the analysis. This is a consequence of the probabilistic nature of the features. One way of solving this issue would be to make n big enough. Instead, we opt to have a fixed n and reduce the degrees of freedom of the models by fixing the parameters shared across all models. The rest of the experimental setup matches the one from App. B. Additionally, we use the Adam optimizer (Kingma & Ba, 2015) with a learning rate of 0.05, and we train for 100.

Results are summarized in Fig. 5. The top row contains the node classification performance for every model, while the bottom row contains the learned values of λ_1 (solid line) and λ_2 (dashed line) with L-CAT. The two left-most columns of Fig. 5 show the results for the hard and easy regimes, respectively, as we vary the noise level q. In the hard regime, we see rather noisy results. Still, the behaviour is similar to that of Fig. 4: the performance of CAT degrades as we increase q. We also observe that, on average, CAT outperforms GAT. In this case, we observe that L-CAT achieves similar performance as CAT, which can be explained by inspecting the learned values of lambda in the bottom row. We observe that $\lambda_1 = 1$ and $\lambda_2 \geq 0.5$ on average for all values of q. This indicates that L-CAT is closer to CAT than to GAT. When the distance between the means is large enough (i.e., $\|\boldsymbol{\mu}\| = 4.3$), we see that GAT achieves perfect results independently of q while the performance of CAT deteriorates with large values of q, the same trend as in Fig. 4. Remarkably, we observe that L-CAT also achieves perfect results independently of q. If we inspect the lambda values, we first see that $\lambda = 1$ for all q, thus the interpolation happens between CAT and GAT. Looking at the values of λ_2, we observe that, for small values of q, λ_2 is pretty noisy, which is expected since any solution achieves perfect performance. Interestingly, we have that $\lambda_2 = 0$ for large values of q, with negligible variance. This indicates that L-CAT learns that it must behave like GAT in order to perform well.

For the second set of experiments (two right-most columns of Fig. 5), we fix q and sweep $\|\boldsymbol{\mu}\|$ like we did in Fig. 4. Here, we observe a similar trend: for both values of q, there exists a transition in the accuracy of both GAT and CAT as a function of $\|\boldsymbol{\mu}\|$. Yet once again, we observe that L-CAT consistently achieves a similar performance to the best-performing model in every situation.

C Dataset description

We present further details about the datasets used in our experiments, summarized in Table 5. All datasets are undirected (or transformed to undirected otherwise) and transductive.

The upper rows of the table refer to datasets used in §6.2 taken from the PyTorch Geometric framework.[4] The following paragraphs present a short description of such datasets.

Amazon Computers & Photos are datasets taken from Shchur et al. (2018), in which nodes represent products, and edges indicate that the products are usually bought together. The node features are a Bag of Words (BoW) representation of the product reviews. The task is to predict the category of the products.

GitHub is a dataset introduced in Rozemberczki et al. (2021), in which nodes correspond to developers, and edges indicate mutual follow relationship. Node features are embeddings extracted from the developer's starred repositories and profile information (e.g., location or employer). The task is to infer whether a node relates to web or machine learning development.

FacebookPagePage is a dataset introduced in Rozemberczki et al. (2021), where nodes are Facebook pages, and edges imply mutual likes between the pages. Nodes features are text embeddings extracted from the pages' description. The task consist on identifying the page's category.

TwitchEN is a dataset introduced in Rozemberczki et al. (2021). Here, nodes correspond to Twitch gamers, and links reveal mutual friendship. Node features are an embedding of

[4] https://pytorch-geometric.readthedocs.io/en/latest/modules/datasets.html

games liked, location, and streaming habits. The task is to infer if a gamer uses explicit content.

Coauthor Physics & CS are datasets introduced in Shchur et al. (2018). In this case, nodes represent authors which are connected with an edge if they have co-authored a paper. Node features are BoW representations of the keywords of the author's papers. The task consist on mapping each author to their corresponding field of study.

DBLP is a dataset introduced in Bojchevski & Günnemann (2018) that represents a citation network. In this dataset, nodes represent papers and edges correspond to citations. Node features are BoW representations of the keywords of the papers. The task is to predict the research area of the papers.

PubMed, Cora & CiteSeer are citation networks introduced in Yang et al. (2016). Nodes represent documents, and edges refer to citations between the documents. Node features are BoW representations of the documents. The task is to infer the topic of the documents.

The bottom rows of Table 5 refer to the datasets from Open Graph Benchmark (OGB) (Hu et al., 2020a) [5] used in §6.3. We include a short description of them in the paragraphs below.

ogbn-arxiv is a citation network of computer science papers in arXiv (Wang et al., 2020). Nodes represent papers, and directed edges refer to citations among them. Node features are embeddings of the title and abstract of the papers. The task is to predict the research area of the nodes.

ogbn-products contains a co-purchasing network (Bhatia et al., 2016). Nodes represent products, and links are present whenever two products are bought together. Node features are embeddings of a BoW representation of the product description. The task is to infer the category of the products.

ogbn-mag is a heterogeneous network formed from a subgraph of the Microsoft Academic Graph (MAG) (Wang et al., 2020). Nodes can belong to one of these four types: authors, papers, institutions and fields of study. Moreover, directed edges belong to one of the following categories: "author is affiliated with an institution," "author has written a paper," "paper cites a paper," and "paper belongs to a research area." Only nodes that are papers contain node features, which are a text embedding of the document content. The task is to predict the venue of the nodes that are papers.

ogbn-proteins is a network whose nodes represent proteins and edges indicate different types of associations among them. This dataset does not contain node features. The tasks are to predict multiple protein functions, each of them being a binary classification problem.

D Real data experiments

D.1 Experimental details

Computational resources. We used CPU cores to run this set of experiments. In particular, for each trial, we used 2 CPU cores and up to 16 GB of memory. We ran the experiments in parallel using a shared cluster with 10000 CPU cores approximately.

General experimental setup. As mentioned in §6.2, we repeat all experiments 10 times, which correspond to 10 different random initialization of the parameters of the GNNs. In all cases, we choose the model parameters with the best validation performance during training. In order to run the experiments and collect the results, we used the GraphGym framework (You et al., 2020), which includes the data processing and loading of the datasets, as well as the evaluation and collection of the results. We split the datasets in 70 % training, 15 % validation, and 15 % test.

We cross-validate the number of message-passing layers in the network $(2, 3, 4)$, as well as the learning rate ($[0.01, 0.005]$). Then, we report the results of the best validation error among the 4 possible combinations. However, in practice we found the best performance always to

[5] https://ogb.stanford.edu/docs/nodeprop

Table 5: Dataset statistics. On the top part of the table, we show the datasets used in §6.2. On the bottom part of the table, we show the datasets used in §6.3.

Name	#Nodes	#Edges	Avg. degree	#Node feats.	#Edge feats.	#Tasks	Task Type
AmazonComp.	13,752	491,722	35.76	767	-	1	10-class clf.
AmazonPhoto	7,650	238,162	31.13	745	-	1	8-class clf.
GitHub	37,700	578,006	15.33	128	-	1	Binary clf.
FacebookP.	22,470	342,004	15.22	128	-	1	4-class clf.
CoauthorPh.	34,493	495,924	14.38	8415	-	1	5-class clf.
TwitchEN	7,126	77,774	10.91	128	-	1	Binary clf.
CoauthorCS	18,333	163,788	8.93	6805	-	1	15-class clf.
DBLP	17,716	105,734	5.97	1639	-	1	4-class clf.
PubMed	19,717	88,648	4.50	500	-	1	3-class clf.
Cora	2,708	10,556	3.90	1433	-	1	7-class clf.
CiteSeer	3,327	9,104	2.74	3703	-	1	6-class clf.
ogbn-arxiv	169,343	1,166,243	6.89	128	-	1	40-class clf.
ogbn-products	2,449,029	123,718,280	50.52	100	-	1	47-class clf.
ogbn-mag	1,939,743	21,111,007	18.61	128	4	1	349-class clf.
ogbn-proteins	132,534	79,122,504	597.00	-	8	112	Multi-task

use 4 message-passing layers, and thus the only difference in configuration lies in the learning rate.

We use residual connections between the GNN layers, 4 heads in the attention models, and the Parametric ReLU (PReLU) (He et al., 2015) as the nonlinear activation function. We do not use batch normalization (Ioffe & Szegedy, 2015), nor dropout (Srivastava et al., 2014). We use the Adam optimizer (Kingma & Ba, 2015) with $\beta = (0.9, 0.999)$, and an exponential learning-rate scheduler with $\gamma = 0.998$. We train all the models for 2500 epochs. Importantly, we do not use weight decay, since this will bias the solution towards $\lambda_1 = 0$ and $\lambda_2 = 1$.

We use the Pytorch Geometric (Fey & Lenssen, 2019) implementation of L-CAT for all experiments, switching between models by properly by setting λ_1 and λ_2. We parametrize λ_1 and λ_2 as free-parameters in log-space that pass through a sigmoid function—i.e., $\texttt{sigmoid}(10^x)$—such that they are constrained to the unit interval, and they are learned quickly.

D.2 Additional results

Table 6 shows the results presented in the main paper (with the addition of a dense feed-forward network), while Table 7 presents the results for the remaining datasets, with smaller average degree.

If we focus on Table 7, we observe that all models perform equally well, yet in a few cases CAT and L-CAT are significantly better than the baselines—e.g., L-CATv2 in *CoauthorCS*, or L-CAT in *Cora*. Following a similar discussion as the one presented in the main paper, these results indicates that L-CAT achieves similar or better performance than baseline models and thus, should be the preferred architecture.

Competitive performance without the graph. We also include in Tables 6 and 7 the performance of a feed-forward network, referred to as Dense (first row). Note that the only data available to this model are the node features, and thus no graph information is provided. Therefore, we should expect a significant drop in performance, which indeed happens for some datasets such as *Amazon Computers* ($\approx 7\%$ drop), *FacebookPagePage* ($\approx 20\%$ drop), *DBLP* ($\approx 9\%$ drop) and *Cora* ($\approx 14\%$ drop). Still, we found that for other commonly used datasets the performance is similar, e.g., *Coauthor Physics* and *PubMed*; or *it is even better CoauthorCS*. These results manifest the importance of a proper benchmarking, and of carefully considering the datasets used to evaluate GNN models.

Table 6: Test accuracy (%) of the considered convolution and attention models for different datasets (sorted by their average node degree), and averaged over ten runs. Bold numbers are statistically different to their baseline model ($\alpha = 0.05$). Best average performance is underlined.

Dataset	*Amazon Computers*	*Amazon Photo*	*GitHub*	*Facebook PagePage*	*Coauthor Physics*	*TwitchEN*
Avg. Deg.	35.76	31.13	15.33	15.22	14.38	10.91
Dense	83.73 ± 0.34	91.74 ± 0.46	81.21 ± 0.30	75.89 ± 0.66	95.41 ± 0.14	56.26 ± 1.74
GCN	90.59 ± 0.36	95.13 ± 0.57	84.13 ± 0.44	94.76 ± 0.19	96.36 ± 0.10	57.83 ± 1.13
GAT	89.59 ± 0.61	94.02 ± 0.66	83.31 ± 0.18	94.16 ± 0.48	96.36 ± 0.10	57.59 ± 1.20
CAT	**90.58 ± 0.40**	**94.77 ± 0.47**	**84.11 ± 0.66**	**94.71 ± 0.30**	96.40 ± 0.10	58.09 ± 1.61
L-CAT	**90.34 ± 0.47**	**94.93 ± 0.37**	**84.05 ± 0.70**	**94.81 ± 0.25**	96.35 ± 0.10	57.88 ± 2.07
GATv2	89.49 ± 0.53	93.47 ± 0.62	82.92 ± 0.45	93.44 ± 0.30	96.24 ± 0.19	57.70 ± 1.17
CATv2	**90.44 ± 0.46**	**94.81 ± 0.55**	**84.10 ± 0.88**	**94.27 ± 0.31**	96.34 ± 0.12	57.99 ± 2.02
L-CATv2	**90.33 ± 0.44**	**94.79 ± 0.61**	**84.31 ± 0.59**	**94.44 ± 0.39**	96.29 ± 0.13	57.89 ± 1.53

Table 7: Test accuracy (%) of the considered convolution and attention models for different datasets (sorted by their average node degree), and averaged over ten runs. Bold numbers are statistically different to their baseline model ($\alpha = 0.05$). Best average performance is underlined.

Dataset	*CoauthorCS*	*DBLP*	*PubMed*	*Cora*	*CiteSeer*
Avg. Deg.	8.93	5.97	4.5	3.9	2.74
Dense	94.88 ± 0.21	75.46 ± 0.27	88.13 ± 0.33	72.75 ± 1.72	73.02 ± 1.01
GCN	93.85 ± 0.23	84.18 ± 0.40	88.50 ± 0.18	86.68 ± 0.78	75.76 ± 1.09
GAT	93.80 ± 0.38	84.15 ± 0.39	88.62 ± 0.18	85.95 ± 0.95	75.40 ± 1.43
CAT	93.70 ± 0.31	84.10 ± 0.29	88.58 ± 0.25	85.85 ± 0.79	75.64 ± 0.91
L-CAT	93.65 ± 0.23	84.13 ± 0.26	88.45 ± 0.32	**86.66 ± 0.87**	75.04 ± 1.12
GATv2	93.19 ± 0.64	84.33 ± 0.18	88.52 ± 0.27	85.65 ± 1.01	75.14 ± 1.20
CATv2	93.51 ± 0.34	**84.15 ± 0.41**	88.54 ± 0.29	85.50 ± 0.94	74.68 ± 1.30
L-CATv2	**93.65 ± 0.20**	84.31 ± 0.31	88.48 ± 0.24	85.75 ± 0.72	75.04 ± 1.30

E Open Graph Benchmark experiments

E.1 Experimental details

Computational resources. For this set of experiments, we had at our disposal a set of 16 Tesla V100-SXM GPUs with 160 CPU cores, shared among the rest of the department.

Statistical significance. For each CAT and L-CAT model, we highlight significant improvements according to a two-sided paired t-test ($\alpha = 0.05$), with respect to its corresponding baseline model. For example, for L-CATv2 with 8 heads we perform the test with respect to GATv2 with 8 heads.

General experimental setup. As mentioned in §6.3, we repeat all experiments with OGB datasets 5 times. In all cases, we choose the model parameters with the best validation performance during training. Moreover, when we show the results without specifying the number of heads, we take the model with the best validation error among the two models with 1 and 8 heads.

We use the same implementation of L-CAT for all experiments, switching between models by properly setting λ_1 and λ_2. Experiments on *arxiv*, *mag*, *products* use a version of L-CAT implemented in Pytorch Geometric (Fey & Lenssen, 2019). Experiments on *proteins* use a version of L-CAT implemented in DGL (Wang et al., 2019a). We parametrize λ_1 and λ_2 as free-parameters in log-space that pass through a sigmoid function—i.e., $\texttt{sigmoid}(10^x)$—such that they are constrained to the unit interval, and they are learned quickly.

ArXiv. As described in §6.3, we use the example code from the OGB framework (Hu et al., 2020a). The network is composed of 3 GNN layers with a hidden size of 128. We use batch normalization (Ioffe & Szegedy, 2015) and a dropout (Srivastava et al., 2014) of 0.5 between the GNN layers, and Adam (Kingma & Ba, 2015) with a learning rate of 0.01. We use the ReLU as activation function. For the initial experiments, we train for 1500 epochs, while we train for 500 epochs for the noise experiments in §6.3.1. This is justified given the convergence plots in Fig. 2.

MAG. We adapted the official code from (Brody et al., 2022). The network is composed of 2 layers with 128 hidden channels. This time, we use layer normalization (Ba et al., 2016) and a dropout of 0.5 between the layers. Again, we use ReLU as the activation function, and add residual connections to the network. As with *arxiv*, we use Adam (Kingma & Ba, 2015) with learning rate 0.01. We set a batch size of 20000 and train for 100 epochs.

Products. We use the same setup as (Brody et al., 2022), with a network of 3 GNN layers and 128 hidden dimensions. We apply residual connections once again, with a dropout (Srivastava et al., 2014) of 0.5 between layers. This time, we use ELU as the activation function. The batch size is set to 256. Adam (Kingma & Ba, 2015) is again the optimizer in use, this time with a learning rate of 0.001. We train for 100 epochs, although we apply early stopping whenever the validation accuracy stops increasing for more than 10 epochs. Note the training split of this dataset only contains 8 % of the data.

Proteins. We follow once more the setup of (Brody et al., 2022). The network we use has 6 GNN layers of hidden size 64. Dropout (Srivastava et al., 2014) is set to 0.25 between layers, with an input dropout of 0.1. At the beginning of the network, we place a linear layer followed by a ReLU activation to encode the nodes, and a linear layer at the end of the network to predict the class. Moreover, we use batch normalization (Ioffe & Szegedy, 2015) between layers and ReLU as the activation function. We train the model for 1200 epochs at most, with early stopping after not improving for 10 epochs.

E.2 Additional results

We show in Tables 8 to 10 the results of the main paper for the *arxiv*, *mag*, *products* datasets, respectively, without selecting the best configuration for each type of model. That is, we show the results for both number of heads. Note that we already show the full table of results for the *protein* datasets in the main paper (Table 3). All the trends discussed in the main paper hold.

Table 8: Test accuracy on the *arxiv* dataset for attention models using 1 head and 8 heads.

	GCN	GAT	CAT	L-CAT	GATv2	CATv2	L-CATv2
1h	71.58 ± 0.19	71.58 ± 0.15	**72.04 ± 0.20**	**72.00 ± 0.11**	71.70 ± 0.14	**72.02 ± 0.08**	71.96 ± 0.21
8h	–	71.63 ± 0.11	**72.14 ± 0.20**	**71.98 ± 0.08**	71.72 ± 0.24	71.76 ± 0.14	71.91 ± 0.16

Table 9: Test accuracy on the *mag* dataset for attention models using 1 head and 8 heads.

	GCN	GAT	CAT	L-CAT	GATv2	CATv2	L-CATv2
1h	32.77 ± 0.36	32.35 ± 0.24	31.98 ± 0.46	32.47 ± 0.38	32.76 ± 0.18	**32.43 ± 0.22**	32.68 ± 0.50
8h	–	32.15 ± 0.31	**31.58 ± 0.22**	32.49 ± 0.21	32.85 ± 0.21	**32.34 ± 0.18**	**32.38 ± 0.28**

Table 10: Test accuracy on the *products* dataset for attention models using 1 head and 8 heads.

	GCN	GAT	CAT	L-CAT	GATv2	CATv2	L-CATv2
1h	74.12 ± 1.20	78.53 ± 0.91	**77.38 ± 0.36**	77.19 ± 1.11	73.81 ± 0.39	74.81 ± 1.12	**76.37 ± 0.92**
8h	–	78.23 ± 0.25	**76.63 ± 1.15**	**76.56 ± 0.45**	76.40 ± 0.71	75.20 ± 0.92	**74.70 ± 0.28**

Extrapolation ablation study. Due to page constraints, these results were not added to the main paper. Here, we study two questions. First, how important are λ_1 and λ_2 in the formulation of L-CAT (Eq. 7)? For the sake of completeness, the second question we

attempt to answer here is whether we can obtain similar performance by just interpolating between GCN and GAT (fixing $\lambda_2 = 0$)? Note that we theoretically showed in §§4 and 6.1 that CAT fills up a gap between GCN and GAT, making it preferable in certain settings.

To this end, we repeat the experiments for network-initialization robustness in §6.3.2, since they showed to be the best ones to tell apart the performance across models. We include three additional models: GCN-GAT, which interpolates between GCN and GAT (or GATv2) by learning λ_1 and fixing $\lambda_2 = 0$; CAT-λ_1 which interpolates between GCN and CAT by learning λ_1 and fixing $\lambda_2 = 1$; and CAT-λ_2, which interpolates between GAT and CAT by learning λ_2 and fixing $\lambda_1 = 1$.

Results using GAT and shown in Table 11, and using GATv2 in Table 12. We can observe that GCN-GAT obtains results in between GCN and GAT for all settings, despite being able to interpolate between both layers in each of the six layers of the network. Regarding learning λ_1 and λ_2, we can observe that there is a clear difference between learning boths (L-CAT), and learning a single one. For both attention models, CAT-λ_1 obtains better results than CAT-λ_2 in all settings, but *uniform* with 8 heads. Still, the results of both variants are substantially worse than those of L-CAT in all cases, *demonstrating the importance of learning to interpolate between the three layer types.*

Table 11: Test accuracy on the *proteins* dataset for GCN (Kipf & Welling, 2017) and GAT (Velickovic et al., 2018) attention models using two network initializations, and two numbers of heads (1 and 8).

	GCN	GCN-GAT	GAT	CAT	L-CAT	CAT-λ_1	CAT-λ_2
				uniform initialization			
1h	61.08 ± 2.86	**70.44 ± 1.56**	59.73 ± 4.04	**74.19 ± 0.72**	**77.77 ± 1.44**	**71.97 ± 3.78**	**73.55 ± 1.36**
8h	–	**68.51 ± 0.91**	72.23 ± 3.20	73.60 ± 1.27	**78.85 ± 1.76**	**76.43 ± 2.47**	72.76 ± 2.79
				normal initialization			
1h	80.10 ± 0.61	66.51 ± 3.23	66.38 ± 7.76	73.26 ± 1.84	**78.06 ± 1.40**	**76.77 ± 1.91**	73.39 ± 1.25
8h	–	**69.93 ± 1.93**	79.08 ± 1.64	**74.67 ± 1.29**	79.63 ± 0.79	78.86 ± 1.07	**73.32 ± 1.15**

Table 12: Test accuracy on the *proteins* dataset for GCN (Kipf & Welling, 2017) and GATv2 (Brody et al., 2022) attention models using two network initializations, and two numbers of heads (1 and 8).

	GCN	GCN-GATv2	GATv2	CATv2	L-CATv2	CATv2-λ_1	CATv2-λ_2
				uniform initialization			
1h	61.08 ± 2.86	**69.69 ± 1.59**	59.85 ± 3.05	**64.32 ± 2.61**	**79.08 ± 1.06**	63.24 ± 1.55	**73.41 ± 0.34**
8h	–	**69.94 ± 1.62**	75.21 ± 1.80	74.16 ± 1.45	**78.77 ± 1.09**	**77.61 ± 1.32**	73.96 ± 1.27
				normal initialization			
1h	80.10 ± 0.61	68.54 ± 1.63	69.13 ± 9.48	74.33 ± 1.06	**79.07 ± 1.09**	78.41 ± 0.93	74.07 ± 1.17
8h	–	**68.71 ± 1.96**	78.65 ± 1.61	**73.40 ± 0.62**	79.30 ± 0.55	78.76 ± 1.41	**73.22 ± 0.77**

F Extending L-CAT to other GNN models

Due to their simplicity and popularity, in the manuscript we focus on the simplest form of GCNs, as described in §2. However, we consider important to remark that L-CAT can be effortless extended to a large range of existing GNN models. A more general formulation of a message-passing GNN layer than the one given in Eq. 1 is the following:

$$\tilde{\boldsymbol{h}}_i = f(\boldsymbol{h}'_i) \quad \text{where} \quad \boldsymbol{h}'_i \stackrel{\text{def}}{=} \bigoplus_{j \in N_i^*} \widehat{\gamma}_{ij} M\left(\boldsymbol{h}_i, \boldsymbol{h}_j; \theta_M\right) \ , \tag{13}$$

where $\widehat{\gamma}_{ij}$ is a scalar value, $\bigoplus$ refers to any permutation invariant operation—e.g., sum, mean, maximum, or minimum operations—and M is the message operator, which can be parameterized, and produces a message based on the sender and receiver representations. This formulation comprises most GNN architectures present in the current literature. For example:

- GCN (Kipf & Welling, 2017): $\boldsymbol{h}'_i = \sum_{j \in N_i^*} \widehat{\gamma}_{ij} \boldsymbol{W}_v \boldsymbol{h}_j$ where $\widehat{\gamma}_{ij} = \frac{1}{|N_i^*|}$ as consider in the main paper, or $\widehat{\gamma}_{ij} = \frac{1}{\sqrt{d_j d_i}}$, where d_i is the number of neighbors of node i (including self-loops), if we consider the symmetric normalized adjacency matrix instead.
- GIN (Xu et al., 2019): $\boldsymbol{h}'_i = (1+\varepsilon)\widehat{\gamma}_{ii}\boldsymbol{h}_i + \sum_{j \in N_i} \widehat{\gamma}_{ij} \boldsymbol{h}_j$ with $\widehat{\gamma}_{ij} = 1$.
- PNA (Corso et al., 2020): $\boldsymbol{h}'_i = \bigoplus_{j \in N_i^*} \widehat{\gamma}_{ij} M\left(\boldsymbol{h}_i, \boldsymbol{h}_j; \theta_M\right)$ where $\widehat{\gamma}_{ij} = 1$, M is an multi-layer perceptron, and $\bigoplus$ is a set of permutation invariant operations, e.g., $\bigoplus = [\mu, \sigma, \max, \min]$.
- GCNII (Chen et al., 2020): $\boldsymbol{h}'_i = \left(\alpha \boldsymbol{h}_i^0 + (1-\alpha) \sum_{j \in N_i^*} \widehat{\gamma}_{ij} \boldsymbol{h}_j\right) \left((1-\beta)\boldsymbol{I} + \beta \boldsymbol{W}_v\right)$ where, just as in the GCN case, $\widehat{\gamma}_{ij} = \frac{1}{\sqrt{d_j d_i}}$, and $0 \leq \alpha, \beta \leq 1$.

In all the models above, the values $\widehat{\gamma}_{ij}$ are taken from the adjacency matrix A (whose entries are 1 if there exists an edge between nodes i and j, and 0 otherwise), or a matrix derived from it, e.g., the symmetric normalized adjacency matrix.

Note that the attention coefficients γ_{ij} defined in Eq. 3 can be understood as an attention-equivalent of the adjacency matrix. Indeed, by defining A^{att} as a matrix whose entries are $|N_i^*|\gamma_{ij}$, one can obtain a row-stochastic matrix that can substitute the adjacency matrix of any GNN model. This technique to generalize attention models to GNN variants more complex than a GCN has been successfully applied in prior literature (Wang et al., 2021b).

With this new interpretation of attention-models, the interpolation performed by L-CAT (see Eq. 7) can similarly be re-interpreted. Indeed, L-CAT learns to interpolate between the adjacency matrix A, and an attention-based adjacency matrix A^{att}, which can be produced by either GAT (Eq. 3) or CAT (Eq. 6), depending on the value of λ_2.

F.1 PNA experiments

In Tables 13 and 14, we show the results—for the datasets described in App. C—using L-CAT and CAT in conjunction with the PNA model (Corso et al., 2020). First, we note that standard PNA works quite well in most cases. Second, if we focus on Table 13, we observe that the standard attention models (i.e., PNAGAT and PNAGATv2) perform significantly worse than the other approaches, in particular on datasets with large average degree, e.g., on *Amazon Computers*. Finally, we observe that the L-CAT models (i.e., L-PNACAT and L-PNACATv2) drastically improve the performance of their attention counterparts and achieve similar performance as the PNA model, with lower performance on the *GitHub* and *Facebook* datasets and higher performance on *Cora* and *CiteSeer*.

To keep the same number of parameters, we reuse the PNA weights to compute the attention scores ($\boldsymbol{W}_q = \boldsymbol{W}_k = \boldsymbol{W}_v$). However, this could be detrimental, as the role of $\boldsymbol{W}_v$ is completely different from that of $\boldsymbol{W}_q$ and $\boldsymbol{W}_v$. Tables 15 and 16 show the same experiments as before, but using different parameters to compute the keys and queries (i.e., $\boldsymbol{W}_q = \boldsymbol{W}_k \neq \boldsymbol{W}_v$). We

observe that the increase of parameters generally helps both CAT and L-CAT models, now outperform the base PNA model in some settings.

Table 13: Test accuracy (%) of the PNA (Corso et al., 2020) models for different datasets (sorted by average node degree), averaged over ten runs. Bold numbers are statistically different to their baseline model ($\alpha = 0.05$). Best average performance is underlined.

Dataset	*Amazon Computers*	*Amazon Photo*	*GitHub*	*Facebook PagePage*	*Coauthor Physics*	*TwitchEN*
Avg. Deg.	35.76	31.13	15.33	15.22	14.38	10.91
PNA	86.51 ± 1.22	93.23 ± 0.65	82.33 ± 0.51	94.28 ± 0.34	96.09 ± 0.14	59.25 ± 1.19
PNAGAT	57.59 ± 10.19	74.78 ± 8.74	72.77 ± 2.06	71.49 ± 11.23	96.05 ± 0.25	54.22 ± 3.02
PNACAT	**81.48 ± 3.81**	**91.73 ± 1.24**	75.55 ± 3.33	**93.10 ± 0.41**	96.16 ± 0.15	**59.11 ± 1.94**
L-PNACAT	**86.45 ± 1.42**	**92.76 ± 0.74**	**78.74 ± 2.91**	**93.59 ± 0.39**	96.24 ± 0.13	**59.12 ± 2.74**
PNAGATv2	36.93 ± 4.07	60.13 ± 4.81	73.93 ± 1.89	58.91 ± 3.42	95.61 ± 0.29	54.45 ± 1.60
PNACATv2	**79.08 ± 2.62**	**88.61 ± 3.24**	75.11 ± 2.79	**92.77 ± 0.50**	**96.06 ± 0.18**	**56.72 ± 2.43**
L-PNACATv2	**85.10 ± 1.70**	**92.19 ± 0.55**	**79.79 ± 1.40**	**93.54 ± 0.36**	**96.03 ± 0.19**	**58.19 ± 1.53**

Table 14: Test accuracy (%) of the PNA (Corso et al., 2020) models for different datasets (sorted by average node degree), averaged over ten runs. Bold numbers are statistically different to their baseline model ($\alpha = 0.05$). Best average performance is underlined.

Dataset Avg. Deg.	*CoauthorCS* 8.93	*DBLP* 5.97	*PubMed* 4.5	*Cora* 3.9	*CiteSeer* 2.74
PNA	93.30 ± 0.31	83.37 ± 0.32	88.37 ± 0.73	84.94 ± 1.19	73.92 ± 0.97
PNAGAT	92.46 ± 0.95	83.42 ± 0.39	88.40 ± 0.33	84.67 ± 0.69	74.64 ± 0.82
PNACAT	92.90 ± 0.24	83.35 ± 0.40	88.24 ± 0.30	**85.58 ± 1.00**	74.94 ± 1.68
L-PNACAT	93.11 ± 0.24	83.21 ± 0.55	88.22 ± 0.40	**85.77** ± 1.01	75.08 ± 1.05
PNAGATv2	90.14 ± 0.82	83.37 ± 0.34	88.14 ± 0.45	85.04 ± 0.86	74.50 ± 1.18
PNACATv2	**92.78 ± 0.27**	83.22 ± 0.38	88.28 ± 0.30	85.41 ± 0.98	74.42 ± 1.11
L-PNACATv2	**93.02 ± 0.37**	83.54 ± 0.65	88.23 ± 0.58	85.48 ± 0.98	74.76 ± 1.57

Table 15: Test accuracy (%) of the PNA (Corso et al., 2020) extended models with $\boldsymbol{W}_q = \boldsymbol{W}_k \neq \boldsymbol{W}_v$ for different datasets, averaged over ten runs. Bold numbers are statistically different to their baseline model ($\alpha = 0.05$). Best average performance is underlined.

Dataset	*Amazon Computers*	*Amazon Photo*	*GitHub*	*Facebook PagePage*	*Coauthor Physics*	*TwitchEN*
Avg. Deg.	35.76	31.13	15.33	15.22	14.38	10.91
PNA	86.51 ± 1.22	93.23 ± 0.65	82.33 ± 0.51	94.28 ± 0.34	96.09 ± 0.14	59.25 ± 1.19
PNAGAT	48.65 ± 19.25	68.01 ± 20.32	72.97 ± 1.07	70.17 ± 12.02	96.02 ± 0.34	53.27 ± 2.54
PNACAT	**83.45 ± 2.60**	**91.62 ± 1.30**	75.35 ± 2.71	**93.31 ± 0.55**	96.22 ± 0.13	**59.23 ± 2.25**
L-PNACAT	**87.18 ± 1.22**	**92.79 ± 0.63**	**79.64 ± 2.54**	**93.78 ± 0.39**	**96.31 ± 0.17**	**59.09 ± 2.50**
PNAGATv2	39.49 ± 4.09	62.19 ± 11.30	73.97 ± 1.67	63.00 ± 4.95	95.83 ± 0.36	55.21 ± 1.05
PNACATv2	**81.20 ± 3.63**	**91.32 ± 0.80**	74.57 ± 2.18	**92.98 ± 0.36**	**96.14 ± 0.16**	56.21 ± 2.01
L-PNACATv2	**86.22 ± 0.83**	**92.98 ± 0.89**	**79.78 ± 2.48**	**93.44 ± 0.37**	**96.13 ± 0.12**	**60.26 ± 1.25**

Table 16: Test accuracy (%) of the PNA (Corso et al., 2020) extended models with $\boldsymbol{W}_q = \boldsymbol{W}_k \neq \boldsymbol{W}_v$ for different datasets, averaged over ten runs. Bold numbers are statistically different to their baseline model ($\alpha = 0.05$). Best average performance is underlined.

Dataset Avg. Deg.	*CoauthorCS* 8.93	*DBLP* 5.97	*PubMed* 4.5	*Cora* 3.9	*CiteSeer* 2.74
PNA	93.30 ± 0.31	83.37 ± 0.32	88.37 ± 0.73	84.94 ± 1.19	73.92 ± 0.97
PNAGAT	92.50 ± 0.46	83.22 ± 0.45	88.43 ± 0.29	84.89 ± 1.15	75.76 ± 1.29
PNAGAT	**92.97 ± 0.47**	83.28 ± 0.59	88.27 ± 0.43	85.09 ± 0.70	75.44 ± 1.51
PNAGAT	**93.17 ± 0.30**	83.50 ± 0.29	88.54 ± 0.45	85.63 ± 0.92	75.22 ± 1.12
PNAGATv2	90.00 ± 1.01	83.40 ± 0.48	88.14 ± 0.31	85.16 ± 0.91	76.14 ± 1.33
PNAGATv2	**92.74 ± 0.20**	**83.05 ± 0.49**	**88.38 ± 0.31**	85.21 ± 0.83	75.80 ± 1.26
PNAGATv2	**93.02 ± 0.30**	83.24 ± 0.44	88.28 ± 0.35	85.04 ± 0.94	75.80 ± 1.19

Table 17: Test accuracy (%) of the GCNII (Chen et al., 2020) models for different datasets, averaged over ten runs. Bold numbers are statistically different to their baseline model ($\alpha = 0.05$). Best average performance is underlined.

Dataset Avg. Deg.	*Amazon Computers* 35.76	*Amazon Photo* 31.13	*GitHub* 15.33	*Facebook PagePage* 15.22	*Coauthor Physics* 14.38	*TwitchEN* 10.91
GCNII	90.82 ± 0.20	95.51 ± 0.48	84.11 ± 0.76	94.03 ± 0.30	96.58 ± 0.11	60.94 ± 1.66
GCNIIGAT	89.04 ± 0.87	94.74 ± 0.57	82.34 ± 0.64	91.18 ± 0.82	96.69 ± 0.13	57.76 ± 1.76
GCNIICAT	**89.83 ± 0.42**	**95.31 ± 0.25**	**83.15 ± 0.51**	**93.25 ± 0.37**	96.69 ± 0.09	**60.51 ± 1.12**
L-GCNIICAT	**90.03 ± 0.42**	95.23 ± 0.39	**83.50 ± 0.57**	**93.71 ± 0.33**	**96.87 ± 0.14**	**61.14 ± 1.64**
GCNIIGATv2	84.26 ± 2.80	89.23 ± 5.30	81.23 ± 0.45	83.82 ± 1.24	96.14 ± 0.28	56.25 ± 1.56
GCNIICATv2	**89.59 ± 0.45**	**95.03 ± 0.55**	**82.45 ± 0.30**	**92.55 ± 0.52**	**96.50 ± 0.09**	**59.04 ± 1.49**
L-GCNIICATv2	**89.81 ± 0.48**	**95.24 ± 0.35**	**83.05 ± 0.49**	**93.68 ± 0.35**	**96.75 ± 0.11**	**61.10 ± 1.11**

F.2 GCNII experiments

Similarly, we have run the experiments from §6.2, this time combining GCNII (Chen et al., 2020) with GAT, CAT, and L-CAT as explained above. Results are shown in Tables 17 and 18, in which we can observe that the baseline model obtains the best results so far in the manuscript (in comparison with both GCN and PNA). And just as before, we observe again that CAT and L-CAT always improve with respect to their base models, staying on par with the baseline GNCII model and, sometimes, even outperforming the baseline model on average (e.g., *Coauthor Physics*, *TwitchEN*). As with the experiments for PNA, Tables 19 and 20 shows the results when the attention matrices are different from the value matrix ($\boldsymbol{W}_q = \boldsymbol{W}_k \neq \boldsymbol{W}_v$). We can similarly observe that most of the results are improved with the additional parameters, beating the baseline model in different datasets.

Table 18: Test accuracy (%) of the GCNII (Chen et al., 2020) models for different datasets, averaged over ten runs. Bold numbers are statistically different to their baseline model ($\alpha = 0.05$). Best average performance is underlined.

Dataset Avg. Deg.	*CoauthorCS* 8.93	*DBLP* 5.97	*PubMed* 4.5	*Cora* 3.9	*CiteSeer* 2.74
GCNII	95.36 ± 0.18	83.86 ± 0.14	89.05 ± 0.28	86.49 ± 0.79	76.46 ± 0.71
GCNIIGAT	95.32 ± 0.27	83.45 ± 0.60	88.24 ± 0.34	85.72 ± 1.05	75.78 ± 0.77
GCNIICAT	95.12 ± 0.25	83.86 ± 0.23	88.65 ± 0.40	85.51 ± 0.95	**76.60 ± 0.50**
L-GCNIICAT	95.30 ± 0.29	83.76 ± 0.26	**88.72 ± 0.35**	85.48 ± 0.96	**76.76 ± 0.51**
GCNIIGATv2	93.37 ± 0.73	83.70 ± 0.42	88.49 ± 0.34	85.82 ± 1.35	75.98 ± 0.71
GCNIICATv2	**95.01 ± 0.32**	83.93 ± 0.36	88.60 ± 0.27	85.72 ± 1.03	76.62 ± 0.63
L-GCNIICATv2	**95.29 ± 0.23**	83.93 ± 0.41	**89.12 ± 0.35**	85.73 ± 0.88	76.22 ± 0.98

Table 19: Test accuracy (%) of the GCNII (Chen et al., 2020) extended models with $\boldsymbol{W}_q = \boldsymbol{W}_k \neq \boldsymbol{W}_v$ for different datasets, averaged over ten runs. Bold numbers are statistically different to their baseline model ($\alpha = 0.05$). Best average performance is underlined.

Dataset	*Amazon Computers*	*Amazon Photo*	*GitHub*	*Facebook PagePage*	*Coauthor Physics*	*TwitchEN*
Avg. Deg.	35.76	31.13	15.33	15.22	14.38	10.91
GCNII	90.82 ± 0.20	95.51 ± 0.48	84.11 ± 0.76	94.03 ± 0.30	96.58 ± 0.11	60.94 ± 1.66
GCNIIGAT	89.24 ± 0.59	94.66 ± 0.59	82.45 ± 0.65	90.90 ± 0.71	96.90 ± 0.16	58.12 ± 2.02
GCNIICAT	**89.94 ± 0.40**	95.03 ± 0.37	**83.12 ± 0.37**	**93.39 ± 0.31**	**96.59 ± 0.07**	59.60 ± 0.76
L-GCNIICAT	**90.35 ± 0.46**	**95.53 ± 0.35**	**83.48 ± 0.47**	**93.63 ± 0.39**	**96.80 ± 0.09**	**60.77 ± 2.15**
GCNIIGATv2	85.70 ± 2.58	91.24 ± 2.43	81.43 ± 0.39	84.59 ± 0.79	96.55 ± 0.20	55.23 ± 2.15
GCNIICATv2	**89.56 ± 0.67**	**95.46 ± 0.54**	**82.50 ± 0.47**	**93.04 ± 0.40**	96.55 ± 0.10	**59.57 ± 1.45**
L-GCNIICATv2	**90.24 ± 0.32**	**95.53 ± 0.28**	**83.34 ± 0.45**	**93.67 ± 0.47**	96.73 ± 0.14	**60.65 ± 1.13**

Table 20: Test accuracy (%) of the GCNII (Chen et al., 2020) extended models with $\boldsymbol{W}_q = \boldsymbol{W}_k \neq \boldsymbol{W}_v$ for different datasets, averaged over ten runs. Bold numbers are statistically different to their baseline model ($\alpha = 0.05$). Best average performance is underlined.

Dataset	*CoauthorCS*	*DBLP*	*PubMed*	*Cora*	*CiteSeer*
Avg. Deg.	8.93	5.97	4.5	3.9	2.74
GCNII	95.36 ± 0.18	83.86 ± 0.14	89.05 ± 0.28	86.49 ± 0.79	76.46 ± 0.71
GCNIIGAT	95.36 ± 0.20	83.60 ± 0.32	88.33 ± 0.35	85.26 ± 1.19	76.30 ± 0.78
GCNIICAT	**95.20 ± 0.12**	**83.89 ± 0.29**	88.45 ± 0.29	86.44 ± 1.22	76.70 ± 0.60
L-GCNIICAT	95.47 ± 0.16	83.70 ± 0.40	88.08 ± 0.47	86.02 ± 1.43	76.54 ± 0.59
GCNIIGATv2	93.97 ± 0.57	83.67 ± 0.24	88.24 ± 0.16	85.28 ± 1.11	76.58 ± 0.64
GCNIICATv2	**95.05 ± 0.33**	83.78 ± 0.35	88.35 ± 0.34	**86.49 ± 0.90**	**75.28 ± 0.84**
L-GCNIICATv2	**95.45 ± 0.18**	83.86 ± 0.24	88.06 ± 0.32	86.49 ± 1.31	76.80 ± 0.42

VACA: Designing Variational Graph Autoencoders for Causal Queries

Pablo Sánchez-Martín [*1,2], **Miriam Rateike** [*1,2], **Isabel Valera** [2]

[1] Max Planck Institute for Intelligent Systems, Tübingen, Germany
[2] Department of Computer Science of Saarland University, Saarbrücken, Germany
psanchez@tue.mpg.de, mrateike@tue.mpg.de, ivalera@cs.uni-saarland.de

Abstract

In this paper, we introduce VACA, a novel class of variational graph autoencoders for causal inference in the absence of hidden confounders, when only observational data and the causal graph are available. Without making any parametric assumptions, VACA mimics the necessary properties of a *Structural Causal Model* (SCM) to provide a flexible and practical framework for approximating interventions (*do-operator*) and *abduction-action-prediction* steps. As a result, and as shown by our empirical results, VACA accurately approximates the interventional and counterfactual distributions on diverse SCMs. Finally, we apply VACA to evaluate counterfactual fairness in fair classification problems, as well as to learn fair classifiers without compromising performance.

1 Introduction

Graph Neural Networks (GNNs) are a powerful tool for graph representation learning and have been proven to excel in practical complex problems like neural machine translation (Bastings et al. 2017), traffic forecasting (Derrow-Pinion et al. 2021; Yu, Yin, and Zhu 2018) or drug discovery (Gilmer et al. 2017).

In this work, we investigate to which extent the inductive bias of GNNs–encoding the causal graph information–can be exploited to answer interventional and counterfactual queries. More specific, to approximate the interventional and counterfactual distributions induced by interventions on a casual model. To this end, we assume i) causal sufficiency–i.e., absence of hidden confounders; and, ii) access to observational data and the true causal graph. We stress that the causal graph can often be inferred from expert knowledge (Zheng and Kleinberg 2019) or via one of the approaches for causal discovery (Glymour, Zhang, and Spirtes 2019; Vowels, Camgoz, and Bowden 2021). With this analysis we aim to complement the concurrent line of research that theoretically studies the use of Neural Networks (NN) (Xia et al. 2021), and more recently GNNs (Zečević et al. 2021), for causal inference.

To this end, we describe the architectural design conditions that a variational graph autoencoder (VGAE)–as a density estimator that leverages a priori graph structure–must fulfill so that it can approximate causal interventions (*do-operator*) and *abduction-action-prediction* steps (Pearl 2009b). The resulting Variational Causal Graph Autoencoder, referred to as VACA, enables *approximating* the observational, interventional and counterfactual distributions induced by a causal model with unknown structural equations. We remark that parametric assumptions on the structural causal equations are in general not testable, may thus not hold in practice (Peters, Janzing, and Schölkopf 2017) and may lead to inaccurate results, if misspecified. VACA addresses this limitation by including uncertainty, i.e., a probabilistic model, in the estimation of the causal-parent relationships.

We show in extensive synthetic experiments that VACA outperforms competing methods (Karimi et al. 2020; Khemakhem et al. 2021) on complex datasets at estimating not only the mean of the interventional/counterfactual distribution (as in previous work), but also the overall distribution (measured in terms of Maximum Mean Discrepancy (Gretton et al. 2012)). Finally, we show a practical use-case in which VACA is used to assess counterfactual fairness of different classifiers trained on the real-world German Credit dataset (Dua and Graff 2017a), as well as to learn counterfactually fair classifiers without compromising performance.

Related Work

Deep generative models are enjoying increasing attention for causal queries in complex data (Moraffah et al. 2020; Parafita and Vitria 2019a). Existing approaches for causal inference focus on i) estimating the Average Treatment Effect (ATE)–a specific type of group-level causal queries–by assuming a fixed causal graph that includes a treatment variable (Kim et al. 2021; Louizos et al. 2017; Rakesh et al. 2018; Schwab, Linhardt, and Karlen 2018; Vowels, Camgoz, and Bowden 2020; Zhang, Zhang, and Li 2020); ii) discovering and intervening on the causal latent structure of the (e.g., image) data (Kim et al. 2021; Parafita and Vitria 2019a,b; Shen et al. 2020; Yang et al. 2020); or iii) addressing interventional and/or counterfactual queries by fitting a conditional model for each observed variable given its causal parents (Garrido et al. 2021; Karimi et al. 2020; Kocaoglu et al. 2018; Parafita and Vitria 2020; Pawlowski, Coelho de Castro, and Glocker 2020).

[*]These authors contributed equally

Within the scope of causality, GNNs have predominantly been used for causal discovery (Yu et al. 2019; Zhang et al. 2019) and only very recently, concurrent with us, exploited to answer interventional queries (Zečević et al. 2021).

Khemakhem et al. (2021) propose CAREFL, an autoregressive normalizing flow for both causal discovery and inference. The authors focus on (multi-dimensional) bi-variate graphs, but their approach can be extended to more general directed acyclic graphs (DAGs) using e.g., neuronal spline flows (Durkan et al. 2019). However, causal assumptions in a graph are modeled not only by the direction of edges, but also the absence of edges (Pearl 2009a). For the task of causal inference CAREFL is unable to exploit the absence of edges fully as it reduces a causal graph to its causal ordering (which may not be unique). Further, the authors only evaluate interventions in root nodes (which reduces to conditioning on the intervened-upon variable).

Karimi et al. (2020) answer interventional queries by fitting a conditional variational autoencoder (CVAE) to each conditional in the Markov factorization implied by the causal graph. As each observed variable is independently fitted, the mismatch between the true and generated distribution can cause errors that propagate to the distribution of its descendants. This can be problematic, especially for long causal paths. Pawlowski, Coelho de Castro, and Glocker (2020) propose an approach similar to Karimi et al. (2020), and additionally propose an approach based on normalizing flows to approximate the causal parent-child effect.

In contrast, VACA leverages i) GNNs to encode the causal graph information (inductive bias), ii) the GNN message passing algorithm to approximate the effect of interventions (*do-operator* (Pearl 2009b)) in the causal graph, and iii) jointly optimizes the observational distribution for all observed variables to avoid error propagation along the Markov factorization. We thoroughly evaluate the performance of VACA and compare it with related work, at approximating both interventional and counterfactual distributions induced by interventions on both root and non-root nodes in a wide variety of causal models.

2 Background

In this section, we first provide a brief overview on SCMs and then introduce the main building block of VACA, i.e., variational graph autoencoders.

Structural causal models

An SCM $\mathcal{M} = (p(\mathbf{U}), \tilde{\mathbf{F}})$ determines how a set of d endogenous (observed) random variables $\mathbf{X} := \{X_1, \dots X_d\}$ is generated from a set of exogenous (unobserved) random variables $\mathbf{U} := \{U_1, \dots U_d\}$ with prior distribution $p(\mathbf{U})$ via the set of *structural equations* $\tilde{\mathbf{F}} = \{X_i := \tilde{f}_i\left(\mathbf{X}_{\text{pa}(i)}, U_i\right)\}_{i=1}^{d}$. Here $\mathbf{X}_{\text{pa}(i)}$ refers to the set of variables directly causing X_i, i.e., parents of i. Similarly to (Karimi et al. 2020; Khemakhem et al. 2021; Pearl 2009a), we consider SCMs that are associated with a directed acyclic *causal graph* (although Section 4 relaxes this assumption). We here denote the causal graph by $\mathcal{G} := (\mathcal{V}, \mathcal{E})$, where each node $i \in \mathcal{V}$ corresponds to an endogenous variable X_i. The set of directed edges $(j, i) \in \mathcal{E}$ represent the causal parent-child relationship between endogenous variables (Pearl 2009a), i.e. X_j is a parent of X_i. $\mathcal{E}$ can be represented by the adjacency matrix $\mathbf{A} \in \{0, 1\}^{d \times d}$, such that $A_{ij} = 1$ if $(j, i) \in \mathcal{E}$ and $A_{ij} = 0$, otherwise. We also define the set of neighbors, a.k.a. parents, of node i as $\text{pa}(i) = \mathcal{N}_i = \{j\}_{(j,i)\in\mathcal{E}}$ and $\text{pa}^*(i) := \text{pa}(i) \cup i$.

Given an SCM, there are two types of causal queries of general interest: interventional queries, e.g., "What would happen to the population $\mathbf{X}$, if variable X_i would be set to a fixed value α?", and counterfactual queries, e.g.,"What would have happened to a specific factual sample $\mathbf{x}^F$, had X_i been set to a value α?". In more detail, *interventional queries* aim to evaluate the effect at the population level (*rung 2*) of a specific intervention on, or equivalently manipulations of, a subset of the endogenous variables $\mathcal{I} \subseteq [d] := \{1, \dots, d\}$. Interventions on an SCM $\mathcal{M}$ are often represented with the *do-operator* $do(X_i = \alpha_i)$ (Pearl 2009b) and lead to a modified SCM $\mathcal{M}^{\mathcal{I}}$ which induces a new distribution over the set of endogenous variables $p(\mathbf{X} \mid do(X_i = \alpha_i))$, which is referred to as the *interventional distribution*. In $\mathcal{G}$ an intervention removes incoming edges to node i and sets $X_i = \alpha$ (see Figure 1b). A *counterfactual query* for a given factual instance $\mathbf{x}^F$ aims to estimate what would have happened had $\mathbf{X}_{\mathcal{I}}$ instead taken value $\boldsymbol{\alpha}$. This effect is captured by the *counterfactual distribution* $p(\mathbf{x}^{CF} \mid \mathbf{x}^F, do(\mathbf{X}_{\mathcal{I}} = \boldsymbol{\alpha}))$, which can be computed using the *abduction-action-prediction* procedure by Pearl (2009b). Refer to Section 3 for further details on the computation of the interventional and counterfactual distributions within our framework.

(a) $\mathcal{G}$ without intervention (b) $\mathcal{G}$ with intervention

Figure 1: *triangle* SCM $\mathcal{M} = \{p(\mathbf{U}), \tilde{\mathbf{F}}\}$, $\mathbf{U} \sim p(\mathbf{U})$ with $d = |\mathbf{X}| = 3$ endogenous variables where $X_1 := \tilde{f}_1(U_1)$, $X_2 := \tilde{f}_2(X_1, U_2)$, $X_3 := \tilde{f}_3(X_1, X_2, U_3)$ with (a) the corresponding causal graph $\mathcal{G}$ and (b) the causal graph corresponding to $\mathcal{M}^{\mathcal{I}}$ after intervention $do(X_2 = \alpha)$. Blue (red) arrows highlight the direct (indirect) causal path from X_1 to X_3 (via X_2).

Variational Graph Autoencoder and Graph Neural Networks

Variational Autoencoders (VAEs). VAEs (Kingma and Welling 2014) are powerful latent variable models based on neural networks (NNs) for jointly i) learning expressive density estimators $p(\mathbf{X}) \approx \int p_\theta(\mathbf{X} \mid \mathbf{Z}) p(\mathbf{Z}) d\mathbf{Z}$, where the likelihood function (a.k.a. *decoder*) is parameterized using a NN with parameters θ, and ii) performing approximate posterior inference over the latent variables $\mathbf{Z}$ made possible via a variational distribution (a.k.a. *encoder*) $q_\phi(\mathbf{Z} \mid \mathbf{X})$

parameterized using a NN with parameters ϕ. The parameters θ and ϕ can be learned by maximizing a lower bound on the log-evidence (Burda, Grosse, and Salakhutdinov 2016; Nowozin 2018; Rainforth et al. 2018; Tucker et al. 2018).

Variational Graph Autoencoders (VGAEs). Kipf and Welling (2016) extend VAEs to account for prior graph structure information on the data (Yu et al. 2019). VGAEs define a (potentially multidimensional) latent variable Z_i per observed variable X_i, i.e., $\mathbf{Z} := \{Z_1, \dots, Z_d\}$. Additionally, VGAEs rely on an adjacency matrix $\mathbf{A}$, which is used by two GNNs, one for the encoder and one for the decoder, to enforce structure on the posterior approximation $q_\phi(\mathbf{Z} \mid \mathbf{X}, \mathbf{A})$ and the likelihood $p_\theta(\mathbf{X} \mid \mathbf{Z}, \mathbf{A})$. Hence, $\mathbf{A}$–given as prior–determines which variables X_i influence Z_j $\forall i, j \in [d]$.

Graph Neural Networks (GNNs). In its most general form, a GNN is a composition of message passing layers (Gilmer et al. 2017), where each layer updates the state of each node in $\mathcal{G}$. In particular, the state of node i at the output of layer l, i.e., $\mathbf{h}_i^l$, is specified as:

$$\mathbf{h}_i^l = f^u\left(\mathbf{h}_i^{l-1}, f^a\left(\{\mathbf{m}_{ij}^l\}_{j\in\mathcal{N}_i}\right); \theta_u^l\right). \tag{1}$$

First, node i receives a message $\mathbf{m}_{ij}^l = f^m(\mathbf{h}_i^{l-1}, \mathbf{h}_j^{l-1}; \theta_m^l)$ from each of its neighbors $j \in \mathcal{N}_i$. Then, these messages are aggregated via f^a. Finally, $\mathbf{h}_i^l$ is computed as a function f^u of the node's previous state $\mathbf{h}_i^{l-1}$ and the aggregated message. Note, if a GNN has N_h hidden layers, then the output for node i depends not only on its direct neighbors $\mathcal{N}_i$, but also on its neighbors up to order $N_h + 1$ (hops). For example, if $N_h = 0$ ($N_h = 1$) then the output for each node only depend on its direct neighbors, i.e., *parents* (2-hop neighbors, i.e., *grand-parents*). For a detailed description of GNNs, please refer to Appendix A.

3 Observational, interventional and counterfactual distributions

In this section, we introduce the observational, interventional and counterfactual distributions (triggered by any intervention of the form $do(\mathbf{X}_\mathcal{I} = \boldsymbol{\alpha})$) that are induced by an SCM $\mathcal{M} = \{p(\mathbf{U}), \tilde{\mathbf{F}}\}$. Specifically, we summarize the main properties of an SCM that will allow us to propose a novel class of VGAEs, namely VACA, to compute accurate estimates of these distributions using observational data and a known causal graph. To this end, we assume the absence of hidden confounders, i.e., we assume that $p(\mathbf{U}) = \prod_{i=1}^d p(U_i)$.

Observational distribution. The SCM $\mathcal{M}$ determines the observational distribution $p(\mathbf{X})$ over the set of endogenous variables $\mathbf{X} = \{X_1, \dots X_d\}$, which satisfies causal factorization (Schölkopf 2019), i.e., $p(\mathbf{X}) = \prod_{i=1}^d p(X_i \mid \mathbf{X}_{\mathrm{pa}(i)})$.

That is, after marginalizing out the exogenous variables $\mathbf{U}$, the distribution of each endogenous variable X_i depends only on its parents, i.e., $\mathbf{X}_{\mathrm{pa}(i)}$. The *observational distribution* can alternatively be written only in terms of the exogenous variables $\mathbf{U}$ as

$$p(\mathbf{X}) = \mathbf{F}_\#[p(\mathbf{U})], \tag{2}$$

i.e., $p(\mathbf{X})$ is the pushforward of $P(\mathbf{U})$ through $\mathbf{F}$. Here $\mathbf{F} : \mathbf{U} \to \mathbf{X}$ corresponds to the set of structural equations, which directly transform the exogenous variables $\mathbf{U}$ into the endogenous variables $\mathbf{X}$. This is equivalent to $\tilde{\mathbf{F}}$, which takes as input both the exogenous variable and the parent (endogenous) variables of a target endogenous variables to compute its value.

Let us denote by $\mathrm{an}(i)$ the set of indexes of the ancestors of i, and $\mathrm{an}^*(i) := \mathrm{an}(i) \cup \{i\}$. Then, the causal factorization induced by $\mathcal{M}$ leads to the following property of $\mathbf{F}(\mathbf{U})$:

Property 1 *Each endogenous variable X_i can be expressed as a function of its exogenous variable U_i and the ones of all its causal ancestors, i.e., $\mathbf{F}(\mathbf{U})$=$\{X_i = f_i(\{U_j\}_{j\in\mathrm{an}^*(i)})\}$. This, together with the causal sufficiency assumption, implies that X_i is statistically independent of $U_j, \forall j \notin \mathrm{an}^*(i)$.*

Interventional distribution. As stated in Section 2, interventions on a set of variables $\mathcal{I}$ can be performed using the *do-operator*, which can be seen as a mapping $do(\mathbf{X}_\mathcal{I} = \boldsymbol{\alpha}) : \mathcal{M} \mapsto \mathcal{M}^\mathcal{I} = (p(\mathbf{U}), \tilde{\mathbf{F}}^\mathcal{I})$ where $\tilde{\mathbf{F}}^\mathcal{I} = \{\tilde{f}_i\}_{i\notin\mathcal{I}} \cup \{\alpha_i\}_{i\in\mathcal{I}}$. As above, we can represent the resulting set of *intervened structural equations* as $\mathbf{F}^\mathcal{I} = \{f_i\}_{i\notin\mathcal{I}} \cup \{\alpha_i\}_{i\in\mathcal{I}}$, and thus write the *interventional distribution* as:

$$p(\mathbf{X} \mid do(\mathbf{X}_\mathcal{I} = \boldsymbol{\alpha})) = \mathbf{F}^\mathcal{I}{}_\#[p(\mathbf{U})]. \tag{3}$$

Property 2 *After an intervention $do(\mathbf{X}_\mathcal{I} = \boldsymbol{\alpha})$ on $\mathcal{M}$, all the causal paths from U_j $\forall j \in \mathrm{an}^*(i)$ to X_i that include an intervened-upon variable in $\mathbf{X}_\mathcal{I}$ (i.e., the causal paths where $\mathbf{X}_\mathcal{I}$ is a mediator) are severed in $\mathbf{F}^\mathcal{I}$, while the rest of causal paths remain untouched.*

The above property is illustrated in Figure 1, where we can observe that after an intervention $do(X_2 = \alpha)$, the indirect causal path (in red) from X_1, and thus from U_1, to X_3 via X_2 is severed, while the direct path (in blue) remains.

Counterfactual distribution. Assuming the SCM $\mathcal{M} = \{p(\mathbf{U}), \tilde{\mathbf{F}}\}$ to be known, the following three steps defined by Pearl (2009a) allow to compute counterfactuals $\mathbf{x}^{CF}$:

i) *Abduction:* infer the values of the exogenous variables $\mathbf{U}$ for a factual sample $\mathbf{x}^F$, i.e., compute $p(\mathbf{U} \mid \mathbf{x}^F)$; ii) *Action:* intervene with $do(\mathbf{X}_\mathcal{I} = \boldsymbol{\alpha}) : \mathcal{M} \mapsto \mathcal{M}^\mathcal{I} = (p(\mathbf{U}), \tilde{\mathbf{F}}^\mathcal{I})$; and iii) *Prediction:* use the posterior distribution $p(\mathbf{U} \mid \mathbf{x}^F)$ and the new structural equations $\tilde{\mathbf{F}}^\mathcal{I}$ to compute $p(\mathbf{x}^{CF} \mid \mathbf{x}^F)$. The prediction step can alternatively be computed using the new set of structural equations $\mathbf{F}^\mathcal{I}$ defined in terms of the exogenous variables $\mathbf{U}$, so that we can write the *counterfactual distribution* as:

$$p(\mathbf{x}^{CF} \mid \mathbf{x}^F, do(\mathbf{X}_\mathcal{I} = \boldsymbol{\alpha})) = \mathbf{F}^\mathcal{I}{}_\#[p(\mathbf{U} \mid \mathbf{x}^F)]. \tag{4}$$

Importantly, the posterior distribution $p(\mathbf{U} \mid \mathbf{x}^F)$ satisfies:

Property 3 *In the abduction step, statistical independence implies that conditioned on the endogenous variables of the factual sample $\mathbf{x}^F$, each exogenous variable U_i is independent of the factual value x_j^F if $j \neq i$ and the variable X_j is not a parent of X_i, i.e., $j \notin \mathrm{pa}^*(i)$.*

4 Variational Causal Autoencoder (VACA)

In this section, we present a novel variational causal graph autoencoder (VACA) to approximate the observational (2), interventional (3) and counterfactual (4) distributions. While the underlying SCM $\mathcal{M}$ is unknown, we assume access to the true causal graph $\mathcal{G}$ and observational data $\{\mathbf{x}_n\}_{n=1}^N$, i.e., i.i.d. samples of the observational distribution induced by $\mathcal{M}$ (in the absence of hidden confounders).

Definition 4.1 *(VACA). Given a causal graph $\mathcal{G}$ over a set of endogenous variables $\mathbf{X} = \{X_1, \dots, X_d\}$, which establishes the set of parents $\text{pa}(i)$ for each variable X_i (including the i-th node), VACA is defined by:*

- *A causal adjacency matrix $\mathbf{A}$, which is a $d \times d$ binary matrix with elements $A_{ij} = 1$ if $j \in \text{pa}^*(i)$, i.e., when $i = j$ or j is a parent of i. Otherwise, $A_{ij} = 0$.*
- *A prior distribution $p(\mathbf{Z}) = \prod_i p(Z_i)$ over the set of latent variables $\mathbf{Z} = \{Z_1, \dots, Z_d\}$.*
- *A decoder $p_\theta(\mathbf{X} \mid \mathbf{Z}, \mathbf{A})$, which is a GNN (parameterized by θ) that takes as input the set of latent variables $\mathbf{Z}$ and the causal adjacency matrix $\mathbf{A}$, and outputs the parameters of the likelihood $p_\theta(\mathbf{X} \mid \mathbf{Z}, \mathbf{A})$.*
- *An encoder $q_\phi(\mathbf{Z} \mid \mathbf{X}, \mathbf{A})$, which is a GNN (parameterized by ϕ) that takes as input the endogenous variables $\mathbf{X}$ and the causal adjacency matrix $\mathbf{A}$, and outputs the parameters of the posterior approximation $q_\phi(\mathbf{Z} \mid \mathbf{X}, \mathbf{A})$.*

Next, we discuss how to design VACA such that it is able to capture the observational, interventional, and counterfactual distribution induced by an unknown SCM. Importantly, we derive the necessary conditions on the design of both the encoder and decoder GNNs such that VACA can mimic the SCM properties introduced in Section 3.

Observational distribution

VACA approximates the *observational distribution* in (2) using the generative model as

$$p(\mathbf{X}) \approx \int p_\theta(\mathbf{X} \mid \mathbf{Z}, \mathbf{A}) p(\mathbf{Z}) d\mathbf{Z}, \tag{5}$$

where $p_\theta(\mathbf{X} \mid \mathbf{Z}, \mathbf{A}) = \prod_{i=1}^d p_\theta(X_i \mid \mathbf{Z}, \mathbf{A})$. Figure 3a depicts this generative process.

Relationship between Z and U. When comparing (5) with the true observational distribution in (2), we observe that the latent variables $\mathbf{Z}$ play a similar role to the exogenous variables $\mathbf{U}$, and the decoder $p_\theta(\mathbf{X} \mid \mathbf{Z}, \mathbf{A})$ plays a similar role to the structural equations $\mathbf{F}$. We remark that $\mathbf{Z}$ do not need to correspond to the true exogenous variables (i.e., $p(\mathbf{U}) \neq p(\mathbf{Z})$), and thus, the decoder does not aim to approximate the causal structural equations. Yet, we assume that there is one independent latent variable Z_i for every observed variable X_i capturing all the information of X_i that cannot be explained by its parents. Thus, since X_i is in turn a (deterministic) function of its parents $\text{pa}(i)$ and its exogenous variable U_i, the posterior $p(Z_i \mid X_i, \text{pa}(i))$ aims to capture the information that U_i contributes to X_i (i.e., the information of X_i not contributed by its parents). That is–similar to $p(U_i \mid X_i, \text{pa}(i))$, the (true) posterior distribution–$p(Z_i \mid X_i, \text{pa}(i))$ should depend only on X_i and parents $\text{pa}(i)$.

(a) Original (b) Intervened

Figure 2: VACA decoder (a) without and (b) with intervening on X_2. Message passing in the GNN correspond to direct (blue) and indirect (red) causal paths in Figure 1.

Observational noise. VACA has observational noise that is not present in the true SCM, where an observed variable is assumed to be a deterministic transformation of its exogenous variables and parents via the structural equations (SEs). As VACA does not have access to the true SEs (nor to the true distribution of the exogenous variables), the *noise* of the likelihood $p_\theta(\mathbf{X} \mid \mathbf{Z}, \mathbf{A})$ can be interpreted as an estimate of the uncertainty on the estimated observational distribution (due to the uncertainty on the true SCM).

Here, we seek to ensure that the $p(\mathbf{X})$ induced by VACA complies with causal factorization (**Property 1** in Section 3). To that end, the design of the decoder GNN must assure that $p_\theta(X_i \mid \mathbf{Z}, \mathbf{A}) = p_\theta(X_i \mid \mathbf{Z}_{\text{an}^*(i)})$. That is, that X_i depends only on Z_j if $j = i$ or X_j is an ancestor of X_i in the causal graph.

Proposition 1 *(Causal factorization). VACA satisfies causal factorization, $p_\theta(\mathbf{X} \mid \mathbf{Z}, \mathbf{A}) = \prod_i p_{\theta_i}(X_i \mid \mathbf{Z}_{\text{an}^*(i)})$, if and only if the number of hidden layers in the decoder is greater or equal than $\delta - 1$, with δ being the length of the longest shortest path between any two endogenous nodes.*

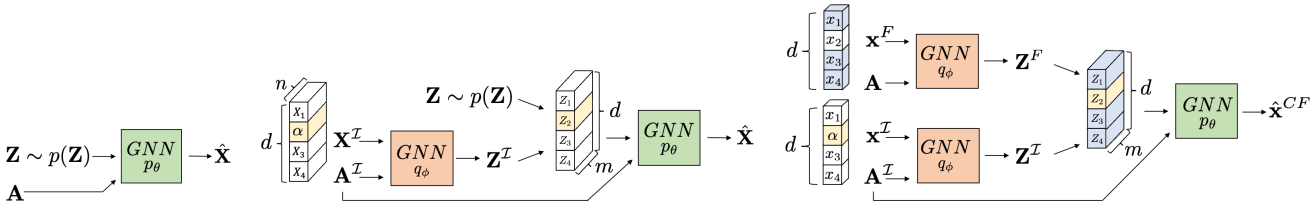
The above proposition (proved in Appendix B) is based on the fact that, in a GNN with N_h hidden layers (and $N_h + 1$ layers in total), the output for the i-th node depends on its neighbors of up to $N_h + 1$ hops. As an example, consider the following *chain* causal graph: $X_1 \to X_2 \to X_3$, such that $\delta = 2$. In the decoder, the first layer yields a hidden representation for the 3-rd node $h_3^1 := f(f(Z_2), Z_3)$ that only depends on Z_2 and Z_3. Thus, we need a second layer for its output $h_3^2 := f(h_2, Z_3) = f(f(f(Z_1), Z_2), Z_3)$ to depend on Z_1 (note that X_1 is an ancestor of X_3).

Interventional distribution

VACA approximates the *interventional distribution* in (3) as (illustrated Figure 2):

$$p(\mathbf{X} \mid do(\mathbf{X}_\mathcal{I} = \boldsymbol{\alpha})) \approx \int\int p_\theta(\mathbf{X} \mid \tilde{\mathbf{Z}}, \tilde{\mathbf{Z}}^\mathcal{I}, \mathbf{A}^\mathcal{I}) \times p(\tilde{\mathbf{Z}}) q_\phi(\tilde{\mathbf{Z}}^\mathcal{I} \mid \mathbf{A}^\mathcal{I}, \mathbf{X}_\mathcal{I}) d\tilde{\mathbf{Z}} d\tilde{\mathbf{Z}}^\mathcal{I}, \tag{6}$$

where $\tilde{\mathbf{Z}}^\mathcal{I} = \{Z_i^\mathcal{I}\}_{i \in \mathcal{I}}$ is the subset of latent variables associated with the intervened-upon variables $\mathbf{X}_\mathcal{I}$, and $\tilde{\mathbf{Z}} = \{Z_i\}_{i \notin \mathcal{I}}$ denotes the subset of latent variables associated with the rest of the observed variables. Importantly, here the *do-operator* is performed on the causal adjacency matrix as $do(X_\mathcal{I} = \boldsymbol{\alpha}) : \mathbf{A} \mapsto \mathbf{A}^\mathcal{I} = \{A_{ij}\}_{\forall i \notin \mathcal{I}, j} \cup \{A_{ij} =$



(a) Observational (b) Interventional (c) Counterfactual

Figure 3: VACA generation of (a) observational, (b) interventional, and (c) counterfactual samples. The 'hat' in $\hat{\mathbf{X}}$ and $\hat{\mathbf{x}}^{CF}$ indicates that they are sample estimates of the true random variables.

$0\}_{\forall i \in \mathcal{I}, j}$. This ensures that X_i for $i \in \mathcal{I}$ is independent of Z_j for all $j \neq i$. Note that in order for (6) to be able to approximate the interventional distribution in (3), an intervention on VACA should satisfy **Property 2**, i.e.:

Proposition 2 *(Causal interventions). VACA captures causal interventions if and only if the number of hidden layers in its decoder is greater than or equal to $\gamma - 1$, with γ being the length of the longest path between any two endogenous nodes in $\mathcal{G}$.*

To illustrate this, Figure 2 depicts how messages are exchanged in a one-hidden-layer decoder GNN corresponding to the causal graph $\mathcal{G}$ in Figure 1 (*triangle* with $\gamma = 2$), both (a) without and (b) with an intervention on X_2. We highlight in blue the direct messages (sent via direct causal path in $\mathcal{G}$), and in red the indirect messages (sent via indirect causal path in $\mathcal{G}$) from Z_1 to X_3. Observe that, similarly to Figure 1, in (a) there is an indirect path (via h_2) from Z_1 to X_3; while in (b) this path is severed. Hence, the hidden layer (h_1, h_2, h_3) allows to distinguish between direct and indirect paths and thus to capture interventional effects. As the condition in Proposition 2 is more restrictive than the one in Proposition 1, VACA is able to approximate the observational and interventional distributions (as empirically validated in Appendix D) if:

Design condition 1 (necessary condition) *The decoder GNN of VACA has at least as many hidden layers as $\gamma - 1$, with γ being the longest directed path in the causal graph $\mathcal{G}$.*

Counterfactual distribution

VACA approximates the *counterfactual distribution* in (4) as (illustrated in Figure 3c):

$$p(\mathbf{x}^{CF} \mid do(\mathbf{X}_{\mathcal{I}} = \boldsymbol{\alpha}), \mathbf{x}^F) \approx \int \int \underbrace{p_\theta(\mathbf{X} \mid \tilde{\mathbf{Z}}^{\mathbf{F}}, \tilde{\mathbf{Z}}^{\mathcal{I}}, \mathbf{A}^{\mathcal{I}}) q_\phi(\tilde{\mathbf{Z}}^{\mathcal{I}} \mid \mathbf{x}^{\mathcal{I}}, \mathbf{A}^{\mathcal{I}})}_{action} \times \underbrace{q_\phi(\tilde{\mathbf{Z}}^{\mathbf{F}} \mid \mathbf{x}^F, \mathbf{A})}_{abduction} d\tilde{\mathbf{Z}}^{\mathcal{I}} d\tilde{\mathbf{Z}}^{\mathbf{F}}, \tag{7}$$

where $\mathbf{x}^F$ represents a sample from $\mathbf{X}$ for which we seek to compute the distribution over counterfactual $\mathbf{x}^{CF}$ and $\tilde{\mathbf{Z}}^{\mathbf{F}} = \{Z_i^F\}_{i \notin \mathcal{I}}$. Note that two different passes of the encoder are necessary: one for the *abduction* step of the factual instance $q_\phi(\tilde{\mathbf{Z}}^{\mathbf{F}} \mid \mathbf{x}^F, \mathbf{A})$; and another one for the *action* step (intervention) $q_\phi(\tilde{\mathbf{Z}}^{\mathcal{I}} \mid \mathbf{x}^{\mathcal{I}}, \mathbf{A}^{\mathcal{I}})$ with $x_i^{\mathcal{I}} = \alpha_i \; \forall i \in \mathcal{I}$ (we remark that the rest of the values in $\mathbf{x}^{\mathcal{I}}$ do not affect the overall counterfactual computation).

We then evaluate the likelihood making sure that the resulting counterfactual sample $\mathbf{x}^{CF}$ only depends on the $\tilde{\mathbf{Z}}^{\mathbf{F}}$ and $\tilde{\mathbf{Z}}^{\mathcal{I}}$. Importantly, in order for VACA to be able to approximate the counterfactual distribution, we need its abduction (and action) step(s) to comply with **Property 3**, i.e.:

Proposition 3 *(Abduction). The abduction step of an observed sample $\mathbf{x} = \{x_1, \ldots, x_d\}$ in VACA satisfies that for all i the posterior of Z_i is independent on the subset $\{x_j\}_{j \notin \mathrm{pa}^*(i)} \subseteq \mathbf{x}$, if and only if the encoder GNN has no hidden layers.*

The above result (proved in Appendix B) can be shown by the message passing algorithm computed by the encoder GNN, and leads to the second design condition of VACA:

Design condition 2 (necessary condition): *The encoder GNN of VACA has no hidden layers.*

In other words, from the definition of the SCM, the posterior distribution of U_i only depends on the parents, i.e $U_i \mid X_{\mathrm{pa}(i)}$. In order for VACA to mimic this property, the GNN that parameterized the encoder contains no hidden layers: in the message passing algorithm, in the k-th iteration (layer), a node depends on its k-hop ancestors; requiring $k = 1$ in the encoder refers to a GNN without hidden layers. Note that while the above condition may look restrictive and limiting the capacity of our encoder, we may choose arbitrarily complex NNs for the message f^m and update f^u functions, as well as one or more aggregation functions f^a, e.g., sum or max, to model the encoder (Corso et al. 2020).

Practical considerations

Next, we briefly discuss practical implementation considerations to handle complex causal models, which often appear in real world applications (Dua and Graff 2017a,b) For further details on VACA implementation, refer to Appendix C.

Heterogeneous causal nodes. So far, we have modeled each endogenous variable X_i as a node in the causal graph $\mathcal{G}$, and thus in the VACA GNNs. Yet, in some application domains the relationships between a subset of k_i variables may be unknown, or they may be affected by hidden confounders. In such cases, we assume that set of k_i variables to be correlated and model them as one multidimensional and potentially heterogeneous node $\mathbf{X}_i = \{X_{i1}, \ldots, X_{ik_i}\}$

that share the same latent random variable Z_i. This allows us to deal with a large variety of graphs in practice.

Heterogeneous endogenous variables. Heterogeneous causal nodes require us to model different functions for each node, i.e. nodes may now contain a mix of continuous/discrete variables. In general GNNs are parametrized such that the parameters of the message function f^m and update function f^u are shared for all the nodes and edges in the graph. However, similar to the structural equations $\mathbf{F}$, we can define a unique set of parameters θ_{mij} for each f^m_{ij} (see (1)), so that we can model a different function for every edge in the causal graph. Further, we can also assume different update functions f^u_i for each node i, by introducing different update parameters θ_{ui}. As a result, VACA fulfills the conditions to be a Neural Causal Model (NCM) Type 2 (Coll. 1 in Zečević et al. (2021)) and thereof can represent the observational, interventional, and counterfactual distributions (Thm.1 and Thm. 3 in Xia et al. (2021)).

Non-identifiability. We highlight that certain counterfactual queries are not identifiable from observational data without making assumptions on the functional relationships even under causal sufficiency (Pearl 2009a). Yet, we expect (as confirmed by our empirical validation) that sufficiently expressive GNNs will lead to accurate approximations of counterfactual queries.

5 Evaluation

In this section, we evaluate the potential of VACA in approximating the outcomes of causal queries and compare it to two competing methods in synthetic experiments. The synthetic setting allows us to have access the true SEs, which is necessary to evaluate interventional distributions and especially counterfactuals. We consider interventions of the form $do(x_i = \alpha_i)$ for several values of α_i on both root and non-root nodes. We compute all results over the same 10 random seeds and report mean and standard deviation. Refer to Appendix E for a complete description of the experimental setup. Moreover, our code is publicly available at GitHub[1].

Datasets. We consider 6 different synthetic causal graphs that differ in the number of nodes d, diameter δ, and longest path γ. Here, we report the results for i) the *collider* ($d = 3$, $\delta = 1$, $\gamma = 1$) with linear (LIN) and non-linear (NLIN) additive noise SEs, ii) the *loan* from Karimi et al. (2020) ($d = 7, \delta = 2, \gamma = 3$), and iii) the *adult* ($d = 11, \delta = 2, \gamma = 3$) graphs. Note that the two latter ones are synthetic versions of the German Credit dataset (Dua and Graff 2017a) and the Adult datasets (Dua and Graff 2017b), respectively. See Appendix E for further details on the graphs and Appendix E for the results with the remaining graphs.

Metrics. We evaluate the observational distribution using the Maximum Mean Discrepancy (MMD) (Gretton et al. 2012) as distance-measure between the true and estimated distributions, i.e., the lower the MMD the better the distributions match. For the interventional distribution, we additionally report the average estimation squared error of the mean (MeanE) and of the standard deviation (StdE) over all descendants of the intervened-upon variables. For the counterfactual distribution we report the mean squared error (MSE) as well as the standard deviation of the squared error (SSE) between the true and the estimated counterfactual value. More details in Appendix E.

Baselines. We compare VACA with MultiCVAE (Karimi et al. 2020) and CAREFL (Khemakhem et al. 2021) described in Section 1. For a fair comparison, all model hyperparameters have been cross-validated using a similar computational budget (see Appendix E). In Table 1, we report for each model and SCM the best configuration according to observational MMD. We also include a time-complexity analysis in Appendix E.

Results. Table 1 summarizes the results. We observe that, in general, MultiCVAE underperforms the other methods. This may be explained by the fact that MultiCVAE trains each node independently, and thus the discrepancy between the true and generated distributions in one node may be amplified in its descendants.

Comparing VACA to CAREFL, we first observe that VACA performs consistently better in terms of observational MMD, i.e., VACA is able to generate observational samples that better resemble the true ones. Second, regarding the interventional distribution, CAREFL does a good job at fitting the mean (i.e., low MeanE). However,VACA performs consistently better both at approximating the standard deviation (i.e., low StdE) and the true samples (i.e., low MMD). This can be explained by VACA i) leveraging the causal graph (contrary to CAREFL that relies on causal ordering), and ii) optimizing the log-evidence in (5) jointly (contrary to the sequential optimization of MultiCVAE). Thus, VACA approximates the distribution as a whole better, which is a desirable property for studying interventions on a population-level rather than just on average. Lastly, in the approximation of counterfactuals we observe that both CAREFL and VACA exhibit similar performance in terms of MSE and SSE. Note however that CAREFL performs exact inference while VACA is built on approximate inference and is trained on a lower bound on the log-evidence. Finding tighter bounds could boost VACA performance.

6 Use case: counterfactual fairness

We finally show two practical use-cases of our method: assessing counterfactual fairness and training counterfactually fair classifiers. We use the public German Credit dataset (Dua and Graff 2017a) and rely on the causal model proposed by Chiappa (2019) with the following random variables $\mathbf{X}$: sensitive feature $S = \{sex\}$, and non-sensitive features $C = \{age\}$, $R = \{$*credit amount*, *repayment history*$\}$ and $H = \{$*checking account*, *savings*, *housing*$\}$. Then, we aim to predict the binary feature $Y = \{$*credit risk*$\}$ from $\mathbf{X}$. See Appendix F for further details.

Counterfactual fairness. Let $S \subset \mathbf{X}$ be a sensitive attribute (e.g., gender), then the counterfactual unfairness (Kusner et al. 2017) of a classifier $h : \mathbf{X} \to Y$ is measured $\forall \mathbf{x}^{CF}, \alpha' \neq \alpha, y$ as:

$$\begin{aligned} uf = \mid & P(h(\mathbf{x}^{CF}) = y \mid do(S = \alpha), \mathbf{x}^F) \\ & - P(h(\mathbf{x}^{CF}) = y \mid do(S = \alpha'), \mathbf{x}^F) \mid \end{aligned} \tag{8}$$

[1] https://github.com/psanch21/VACA

SCM		Model	Obs. MMD	Interventional MMD	Interventional MeanE	Interventional StdE	Counterfactuals MSE	Counterfactuals SSE	Num. params
collider	LIN	MultiCVAE	30.37±8.16	44.70±12.25	13.29±4.78	46.56±2.40	87.41±3.64	65.15±2.83	553
		CAREFL	9.27±1.49	4.86±0.45	0.35±0.08	81.89±1.78	8.11±0.58	7.83±0.55	6420
		VACA	1.50±0.67	1.57±0.41	0.75±0.31	41.99±0.30	9.86±0.74	7.06±0.38	5600
	NLIN	MultiCVAE	28.03±9.12	41.60±12.62	10.49±4.12	46.48±2.43	82.32±2.61	62.05±1.87	553
		CAREFL	10.38±2.00	4.69±0.38	0.19±0.07	80.68±2.08	6.93±0.40	7.15±0.64	4308
		VACA	0.95±0.27	0.97±0.23	0.26±0.12	42.20±0.24	5.01±0.73	4.08±0.54	1805
loan	-	MultiCVAE	90.38±11.31	213.65±5.38	12.24±1.33	65.78±1.13	40.98±0.35	15.12±0.16	33717
		CAREFL	22.10±1.64	27.38±4.07	6.74±4.25	50.13±2.47	11.15±2.57	6.59±0.38	2880
		VACA	2.22±0.25	6.87±0.66	4.35±0.35	3.83±0.08	10.30±0.40	6.41±0.11	30402
adult	-	MultiCVAE	140.15±6.37	155.52±5.93	12.18±2.36	63.52±4.05	39.96±0.36	16.37±0.65	6549
		CAREFL	31.31±1.58	34.31±5.77	12.54±3.17	41.26±3.44	1.23±0.17	3.55±0.90	127420
		VACA	4.51±0.45	12.68±1.95	1.65±0.23	3.37±0.09	5.33±0.27	5.67±0.20	63432

Table 1: Performance of different methods at estimating the observational, interventional and counterfactual distribution of different complex SCMs. Values are multiplied by 100. All models have been cross-validated with a similar computational budget. The number of parameters of the best configuration is shown in the right column.

A classifier is counterfactually fair (*uf* $= 0$), if, given a factual $\mathbf{x}^F$ with sensitive attribute $S = \alpha$, had its sensitive attribute been different $S = \alpha'$, the classifier prediction would remain the same. We can use VACA to generate counterfactual estimates to *audit* the fairness level of a classifier. Following (Kusner et al. 2017), we *audit*: i) a *full* model $h_{\text{full}} : \mathbf{X} \to Y$ that takes as input the complete variable set; ii) an *unaware* model $h_{\text{unaw}} : \mathbf{X} \backslash S \to Y$ that takes as input all variables but the sensitive one; iii) and a *fair* model $h_{\text{fair-x}} : \{X_i | S \notin \text{an}^*(i)\} \to Y$ that takes as input all non-descendant variables of the sensitive attribute. Moreover, we show that we can *learn a fair classifier* $h_{\text{fair-z}} : \mathbf{Z} \backslash Z_S \to Y$, which takes as input the latent variables generated by the VACA encoder without the one of the sensitive attribute Z_S.

Fairness Auditing. Table 2 summarizes the unfairness level and f1-score for a support vector machine (SVM) classifier. See Appendix F for results of a logistic regression classifier. As we do not have access to the true data generation process, we evaluate the *auditing* task by the resulting ranking of the different classifiers according to their unfairness level. Based on the counterfactual generation by VACA the *full* classifier is consistently less fair than the *unaware* and the *fair-x* classifier, respectively. This ranking is consistent with the one in (Kusner et al. 2017).

Fairness Classification Table 2 shows that for the *fair-x* classifier fairness comes at the expense of accuracy compared to the *full* classifier. On the contrary, even though VACA has been trained for representation learning without access to classification labels, *fair-z* is a fair classifier (with comparable fairness level to the *fair-x* one) while keeping the performance comparable to the unfair *full* classifier. VACA, therefore also provides a practical approach to train accurate and fair classifiers.

7 Conclusion, Limitations and Impact

In this work, we have proposed VACA, a variational causal autoencoder based on GNNs that: i) is specially designed to capture the properties of SCMs; ii) inherently handles heterogeneous data; and iii) provides good approximations of interventional and counterfactual distributions as a whole for SCMs of different complexities. As demonstrated by extensive synthetic experiments, VACA provides accurate results for a wide range of interventions in diverse SCMs leading to more consistent results than competing methods (Karimi et al. 2020; Khemakhem et al. 2021). Finally, we have applied VACA for counterfactually fair classification.

Practical limitations. The expressive power of VACA to model complex structural equations, e.g., in domains such as biology (Sachs et al. 2005), is limited by the GNN architectures of the encoder and the decoder. As discussed in the GNN literature (Corso et al. 2020), especially aggregation functions may limit expressiveness. We expect VACA to benefit from advances in the field. Second, long causal paths would require VACA to increase the number of layers in the decoder (see **Design condition 1**). However, the GNNs performance is known to deteriorate with depth (Gallicchio and Micheli 2020; Gu et al. 2020; Li, Han, and Wu 2018).

Social impact. Trusting counterfactuals is of great importance for decision making, e.g. in the political or medical domain. We thus encourage anyone who uses VACA (or any other ML method for causal inference) to fully understand the model assumptions and to verify (up to the possible extend) that they are fulfilled.

Future work. First, it would be important to evaluate the sensitivity of VACA to errors in the assumed causal graph, as well as to the presence of hidden confounders. We plan to extend VACA to handle more complex causal models including, e.g., hidden confounders and non-DAG causal graphs. Second, it would be interesting to perform ablation studies on the limitations of available GNNs architectures (Wu et al. 2020) for the VACA encoder and decoder; as

Metric	full	unaware	fair-x	fair-z
$\uparrow$ *f1*	71.67	69.49	59.50	70.79 ± 5.15
$\downarrow$ *uf*	14.01 ± 2.26	13.27 ± 2.28	0.14 ± 0.02	0.51 ± 0.19

Table 2: Counterfactual unfairness (*uf*) and f1-score (*f1*) of an SVM over 10 VACA seeds. Values multiplied by 100.

well as on how the performance deteriorates as we increase the length of the causal path and thus the required number of hidden layers (Li, Han, and Wu 2018). Finally, it would be intriguing to apply VACA to other causal questions such as privacy-preserving causal inference (Kusner et al. 2016) or explainable machine learning (Karimi et al. 2020).

Acknowledgments

We would like to thank Amir Hossein-Karimi, Adrián Javaloy Bornás, Jonas Kleesen and Maryam Meghdadi Esfahani for helpful feedback and discussions. Moreover, a special thanks to Diego Baptista Theuerkauf for invaluable help with formalizing proofs. Moreover, the authors would like to thank Ilyes Khemakhem for helpful insights in how to generalize their CAREFL approach to arbitrary graphs.

Pablo Sańchez Martín thanks the German Research Foundation through the Cluster of Excellence "Machine Learning – New Perspectives for Science", EXC 2064/1, project number 390727645 and Miriam Rateike thanks the German Federal Ministry of Education and Research (BMBF): Tübingen AI Center, FKZ: 01IS18039B, for generous funding support. The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Pablo Sańchez Martín.

References

Bastings, J.; Titov, I.; Aziz, W.; Marcheggiani, D.; and Sima'an, K. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, volume 3.

Burda, Y.; Grosse, R.; and Salakhutdinov, R. 2016. Importance weighted autoencoders. In *Proceedings of the International Conference onLearning Representations (ICLR)*, volume 4.

Chiappa, S. 2019. Path-specific counterfactual fairness. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33.

Corso, G.; Cavalleri, L.; Beaini, D.; Liò, P.; and Veličković, P. 2020. Principal neighbourhood aggregation for graph nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33.

Derrow-Pinion, A.; She, J.; Wong, D.; Lange, O.; Hester, T.; Perez, L.; Nunkesser, M.; Lee, S.; Guo, X.; Wiltshire, B.; et al. 2021. ETA Prediction with Graph Neural Networks in Google Maps. *arXiv preprint arXiv:2108.11482*.

Dua, D.; and Graff, C. 2017a. UCI Machine Learning Repository.

Dua, D.; and Graff, C. 2017b. UCI Machine Learning Repository.

Durkan, C.; Bekasov, A.; Murray, I.; and Papamakarios, G. 2019. Neural Spline Flows. In *Advances in Neural Information Processing Systems*, volume 32.

Gallicchio, C.; and Micheli, A. 2020. Fast and deep graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34.

Garrido, S.; Borysov, S.; Rich, J.; and Pereira, F. 2021. Estimating causal effects with the neural autoregressive density estimator. *Journal of Causal Inference*, 9.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 34. PMLR.

Glymour, C.; Zhang, K.; and Spirtes, P. 2019. Review of causal discovery methods based on graphical models. *Frontiers in genetics*, 10: 524.

Gretton, A.; Borgwardt, K. M.; Rasch, M. J.; Schölkopf, B.; and Smola, A. 2012. A kernel two-sample test. *The Journal of Machine Learning Research (JMLR)*, 13.

Gu, F.; Chang, H.; Zhu, W.; Sojoudi, S.; and El Ghaoui, L. 2020. Implicit Graph Neural Networks. In *Advances in Neural Information Processing Systems*, volume 33.

Karimi, A.-H.; von Kügelgen, J.; Schölkopf, B.; and Valera, I. 2020. Algorithmic recourse under imperfect causal knowledge: a probabilistic approach. In *Advances in Neural Information Processing Systems*, volume 33, 265–277.

Khemakhem, I.; Monti, R.; Leech, R.; and Hyvarinen, A. 2021. Causal autoregressive flows. In *Proceedings of International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 24. PMLR.

Kim, H.; Shin, S.; Jang, J.; Song, K.; Joo, W.; Kang, W.; and Moon, I.-C. 2021. Counterfactual fairness with disentangled causal effect variational autoencoder. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35.

Kingma, D. P.; and Welling, M. 2014. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, volume 2.

Kipf, T. N.; and Welling, M. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*.

Kocaoglu, M.; Snyder, C.; Dimakis, A. G.; and Vishwanath, S. 2018. CausalGAN: Learning Causal Implicit Generative Models with Adversarial Training. In *Proceedings of the International Conference on Learning Representations (ICLR)*, volume 6.

Kusner, M. J.; Loftus, J.; Russell, C.; and Silva, R. 2017. Counterfactual fairness. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30.

Kusner, M. J.; Sun, Y.; Sridharan, K.; and Weinberger, K. Q. 2016. Private causal inference. In *Proceedings of the Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 19. PMLR.

Li, Q.; Han, Z.; and Wu, X.-M. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Louizos, C.; Shalit, U.; Mooij, J. M.; Sontag, D.; Zemel, R.; and Welling, M. 2017. Causal effect inference with deep latent-variable models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30.

Moraffah, R.; Moraffah, B.; Karami, M.; Raglin, A.; and Liu, H. 2020. CAN: A causal adversarial network for learning observational and interventional distributions. *arXiv preprint arXiv:2008.11376*.

Nowozin, S. 2018. Debiasing evidence approximations: On importance-weighted autoencoders and jackknife variational inference. In *Proceedings of the International Conference on Learning Representations (ICML)*, volume 35. PMLR.

Parafita, A.; and Vitria, J. 2019a. Explaining visual models by causal attribution. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, 4167–4175. IEEE.

Parafita, A.; and Vitria, J. 2019b. Explaining visual models by causal attribution. In *International Conference on Computer Vision Workshop (ICCVW)*.

Parafita, A.; and Vitria, J. 2020. Causal inference with deep causal graphs. *arXiv preprint arXiv:2006.08380*.

Pawlowski, N.; Coelho de Castro, D.; and Glocker, B. 2020. Deep structural causal models for tractable counterfactual inference. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33.

Pearl, J. 2009a. Causal inference in statistics: An overview. *Statistics surveys*, 3.

Pearl, J. 2009b. *Causality*. Cambridge university press.

Peters, J.; Janzing, D.; and Schölkopf, B. 2017. *Elements of causal inference: Foundations and learning algorithms*. The MIT Press.

Rainforth, T.; Kosiorek, A.; Le, T. A.; Maddison, C.; Igl, M.; Wood, F.; and Teh, Y. W. 2018. Tighter variational bounds are not necessarily better. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 35. PMLR.

Rakesh, V.; Guo, R.; Moraffah, R.; Agarwal, N.; and Liu, H. 2018. Linked causal variational autoencoder for inferring paired spillover effects. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*. ACM.

Sachs, K.; Perez, O.; Pe'er, D.; Lauffenburger, D. A.; and Nolan, G. P. 2005. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721): 523–529.

Schölkopf, B. 2019. Causality for machine learning. *arXiv preprint arXiv:1911.10500*.

Schwab, P.; Linhardt, L.; and Karlen, W. 2018. Perfect match: A simple method for learning representations for counterfactual inference with neural networks. *arXiv preprint arXiv:1810.00656*.

Shen, X.; Liu, F.; Dong, H.; Lian, Q.; Chen, Z.; and Zhang, T. 2020. Disentangled generative causal representation learning. *arXiv preprint arXiv:2010.02637*.

Tucker, G.; Lawson, D.; Gu, S.; and Maddison, C. J. 2018. Doubly reparameterized gradient estimators for Monte Carlo objectives. In *International Conference on Learning Representations*.

Vowels, M. J.; Camgoz, N. C.; and Bowden, R. 2020. Targeted VAE: Structured inference and targeted learning for causal parameter estimation. *arXiv preprint arXiv:2009.13472*.

Vowels, M. J.; Camgoz, N. C.; and Bowden, R. 2021. D'ya like DAGs? A Survey on Structure Learning and Causal Discovery. *arXiv preprint arXiv:2103.02582*.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Philip, S. Y. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*.

Xia, K.; Lee, K.-Z.; Bengio, Y.; and Bareinboim, E. 2021. The Causal-Neural Connection: Expressiveness, Learnability, and Inference. *arXiv preprint arXiv:2107.00793*.

Yang, M.; Liu, F.; Chen, Z.; Shen, X.; Hao, J.; and Wang, J. 2020. CausalVAE: Disentangled representation learning via neural structural causal models. *arXiv preprint arXiv:2004.08697*.

Yu, B.; Yin, H.; and Zhu, Z. 2018. Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, volume 27.

Yu, Y.; Chen, J.; Gao, T.; and Yu, M. 2019. DAG-GNN: DAG structure learning with graph neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 36. PMLR.

Zečević, M.; Dhami, D. S.; Veličković, P.; and Kersting, K. 2021. Relating Graph Neural Networks to Structural Causal Models. *arXiv preprint arXiv:2109.04173*.

Zhang, C.; Zhang, K.; and Li, Y. 2020. A Causal View on Robustness of Neural Networks. *Advances in Neural Information Processing Systems*, 33.

Zhang, M.; Jiang, S.; Cui, Z.; Garnett, R.; and Chen, Y. 2019. D-VAE: A variational autoencoder for directed acyclic graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32.

Zheng, M.; and Kleinberg, S. 2019. Using domain knowledge to overcome latent variables in causal inference from time series. In *Proceedings of the Machine Learning for Healthcare Conference (MLHC)*. PMLR.

A Background details on message passing Graph Neural Networks

A *directed graph* with $|\mathcal{V}| = d$ nodes can be represented as $\mathcal{G} := (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the set of nodes and $(j, i) \in \mathcal{E}$ is the set of directed edges from j to i. Additionally, we denote with $\mathbf{X} \in \mathbb{R}^{d \times F_X}$ the features of the nodes (the row index identifies the node, i.e., the i-th row contains the F_X-dimensional features of the node i) Also, the adjacency matrix $\mathbf{A} \in \{0,1\}^{d \times d}$ of $\mathcal{G}$ is defined as $A_{ij} = 1$ if there is an edge from j to i and $A_{ij} = 0$ otherwise. Then, a directed graph can be alternatively represented as $\mathcal{G} = (\mathbf{X}, \mathbf{A})$. Given a graph $\mathcal{G}$, a *Graph Neural Network (GNN) with parameters* θ is a function $f_\theta : \mathbb{R}^{d \times F_X} \times \{0,1\}^{d \times d} \to \mathbb{R}^{d \times F_H}$ that takes into account the graph structure contained in the adjacency matrix $\mathbf{A} \in \{0,1\}^{d \times d}$ and transforms the node features $\mathbf{X}$ into different features $\mathbf{H} \in \mathbb{R}^{d \times F_H}$, i.e., $\mathbf{H} = f(\mathbf{X}, \mathbf{A}; \theta)$ (for readability we consider $f_\theta(\cdot) \equiv f(\cdot; \theta)$). Importantly, at the output of the GNN we have a graph $(\mathbf{H}, \mathbf{A})$ that preserves the structure of the input graph $(\mathbf{X}, \mathbf{A})$.

A GNN based on message passing (Gilmer et al. 2017) is a type of spatial convolution GNNs (Wu et al. 2020) in which information is passed following the message passing process: In each layer l of the GNN each node i receives information from its neighbors $\mathcal{N}_i$, a.k.a. the parents of i. In a message passing GNN, the feature vector of the i-th node at the output of a layer l—i.e., h_i^l—is computed in three steps:

1. `Message`. The *message from node j to node i* is defined as $m_{ij}^l = f^m(h_i^{l-1}, h_j^{l-1}; \theta_m^l)$, where h_i^{l-1} are the features of node i at layer $l-1$, h_j^{l-1} are the features of node j at the previous layer $l-1$, and f^m is a neural network (usually a linear layer) parametrized by θ_m^l.
2. `Aggregator`. The *aggregator* is a function in charge of combining all the incoming messages at each node i into a single message, a.k.a. the aggregated message $M_i^l = f^a(\{m_{ij}^l \mid j \in \mathcal{N}_i\})$. Notice that f^a does not have any parameters. Choices of f^a may be the sum, mean, standard deviation, max or min over the inputs, i.e., messages (Corso et al. 2020).
3. `Update`. The *update function* $h_i^l = f^u(h_i^{l-1}, M_i^l; \theta_u^l)$ takes the aggregated message and the representation of node i at layer $l-1$ and outputs the new representation for node i at layer l. The function f^u is defined as a neural network (usually a linear layer) with parameters θ_u^l.

Putting the three steps together, we obtain the general form of a message passing based GNN layer as $h_i^l = f^u\left(h_i^{l-1}, f^a\left(\{f^m(h_i^{l-1}, h_j^{l-1}; \theta_m^l) \mid j \in \mathcal{N}_i\}\right); \theta_u^l\right)$. Algorithm 1 describes the propagation of information (i.e., messages) in a GNN with L layers.

Algorithm 1: Message passing GNN with L layers decomposed in three operations

Input: A directed graph $\mathcal{G}$ with d nodes, adjacency matrix $\mathbf{A}$ and node features $\mathbf{X}$.
Output: $\mathbf{H} = \{h_i^L\}_{i=1}^d$.

```
h_i^0 = x_i ∀i
for l = 1, ..., L;          // For each layer l
 do
    for i = 1, ..., d;          // For node i
     do
        m_ij^l = f^m(h_i^{l-1}, h_j^{l-1}; θ_m^l) ∀j ∈ N_i;
         // Compute the messages
        M_i^l = f^a({m_ij^l}_{j∈N_i}) ; // Compute the
         aggregated message
        h_i^l = f^u(h_i^{l-1}, M_i^l; θ_u^l) ;      // Compute
         the node features at layer l
    end
end
```

B Proofs

For completeness, this section first formalizes the meaning of causal factorization, interventions and the abduction step in VACA.

VACA causal factorization refers to the factorization of the joint distribution as

$$p_\theta(\mathbf{X} \mid \mathbf{Z}, \mathbf{A}) = \prod_i p_{\theta_i}(X_i; \boldsymbol{\eta}_i),$$

where the likelihood parameters $\boldsymbol{\eta}_i = \boldsymbol{\eta}_i(\mathbf{Z}_{\mathrm{an}^*(i)})$ are a function of all (and only) the features of the ancestors of i and the features of i.

A VACA intervention is performed by removing all the edges towards the intervened node i, such that $\mathcal{N}_i = \varnothing$, while the rest of the edges remains untouched.

In a *VACA abduction step*, the posterior distribution factorizes as

$$q_\phi(\mathbf{Z} \mid \mathbf{X}) = \prod_i q_{\phi_i}(Z_i; \boldsymbol{\eta}_i^{enc}),$$

where the distribution parameters $\boldsymbol{\eta}_i^{enc} = \boldsymbol{\eta}_i^{enc}(\mathbf{X}_{\mathrm{pa}^*(i)})$ are a function of all (and only) the features of node i and the features of its the parents.

Notation. Consider a *causal graph* $\mathcal{G} := (\mathbf{X}, \mathbf{A})$, which is a directed acyclic graph (DAG). Let us define a path of length n from node u to node v in $\mathcal{G}$ as $p(u, v) = (u, w_1, w_2, \ldots, w_{n-1}, v)$, which is an ordered sequence of unique nodes such that i) there exists an edge in $\mathcal{G}$ between concurrent nodes, ii) the first node is u, iii) and the last node is v. We refer to the length of the path as $|p(u, v)|$, i.e., the number of edges in the path, or alternatively, the number of nodes minus one. Let us define $P(u, v)$ as the set of unique paths connecting u to v. Let us define the *shortest path from u to v* as $p^-(u, v)$ (i.e. the path with the minimum number of edges to go from u to v) and its length as $d^-(u, v) = |p^-(u, v)|$. Let us define the *longest path* from u to v as $p^+(u, v)$ (i.e. the path with the maximum number of edges to go from u to v) and its length as $d^+(u, v) = |p^+(u, v)|$. Let us define the *set of ancestors of node i* (i.e., $\mathrm{an}(i)$) as the set of nodes with paths to i, i.e., $\{j \mid |P(j, i)| > 0\}$. As for a GNN, we define the number of hidden layers (total number of layers minus one) as N_h.

Then, we define the *diameter* δ of the graph $\mathcal{G}$ to be the length of the longest shortest path and γ to be the length of the longest path of the graph, which we compute as

$$\delta = \max_{u,v \in \mathcal{G}} d^-(u,v) \quad \text{and} \quad \gamma = \max_{u,v \in \mathcal{G}} d^+(u,v).$$

Lemma 1 *A message passing Graph Neural Network (GNN) has at least N_h hidden layers if and only if the output feature of every node i (i.e., $h_i^{N_h+1}$) receives information from any other node j via paths $p(j,i)$ such that $|p(j,i)| \leq N_h + 1$.*

Proof. **Step 1.** The statement is that the feature of every node i at the output of a GNN with N_h hidden layers (i.e., $h_i^{N_h+1}$) receives information from any other node j via paths $p(j,i)$ such that $|p(j,i)| \leq N_h + 1$. We give a proof by induction on N_h for an arbitrary node i, with input feature to the GNN h_i^0 and output feature $h_i^{N_h+1}$.

Base case: The statement holds for $N_h = 0$. By definition, a message passing GNN with one layer only exchanges messages between neighboring nodes. Hence, i) the output feature of node i is $h_i^1 = f(\{h_i^0\} \cup \{h_j^0 | j \in \mathcal{N}_i\}; \theta)$, which is only a function of the 1-hop ancestors (i.e., parents); ii) information is exchanged via paths that fulfill $|p(j,i)| \leq 1$.

Inductive step: We assume the statement holds for $N_h = k-1$. In this case, i) the output feature of node i is $h_i^k = f(\{h_i^{k-1}\} \cup \{h_j^{k-1} | j \in \mathcal{N}_i\}; \theta)$, which is a function of the k-hop ancestors; ii) information is exchanged via paths that fulfill $|p(j,i)| \leq k$. For $N_h = k$, the output feature of node i is $h_i^{k+1} = f(\{h_i^k\} \cup \{h_j^k | j \in \mathcal{N}_i\}; \theta)$. Since h_j^k is a function of k-hop ancestors of node j, it follows that h_i^{k+1} is a function of the ancestors of h_j^k and on parents of node i, i.e., the output feature of node i is a function of its $(k+1)$-ancestors. Then, it follows that for $N_h = k$, information is exchanged via paths that fulfill $|p(j,i)| \leq k+1$.

Step 2. We assume that the output feature of every node i receives information from any other node j via paths $p(j,i)$ such that $|p(j,i)| \leq N_h + 1$. Then, there exist node i and j such that $|p(j,i)| = N_h + 1$. Then, by definition, a message passing GNN needs at least N_h hidden layers to capture paths $p(j,i)$ with length $|p(j,i)| \leq N_h + 1$. □

Remark. Lemma 1 implies that, for a given GNN with N_h hidden layers, the set of paths through which the output feature of node i is a function of node j is

$$\begin{aligned} P_{\text{GNN}}(j,i) = &\{p(j,i) \mid p(j,i) \in P(j,i) \\ &\text{and } |p(j,i)| \leq N_h + 1\}. \end{aligned} \tag{9}$$

Additionally, if $|P_{\text{GNN}}(j,i)| = \varnothing$ then the output feature of node i is not a function of node j.

Proposition 1 (Causal factorization). *VACA satisfies causal factorization, $p_\theta(\mathbf{X} \mid \mathbf{Z}, \mathbf{A}) = \prod_i p_{\theta_i}(X_i \mid \mathbf{Z}_{\text{an}^*(i)})$, if and only if the number of hidden layers in the decoder is greater or equal than $\delta - 1$, with δ being the length of the longest shortest path between any two endogenous nodes.*

Proof. Consider a causal graph $\mathcal{G} := (\mathbf{X}, \mathbf{E})$ with diameter δ and a GNN decoder with N_h hidden layers. We assume VACA to satisfy causal factorization, i.e., $\boldsymbol{\eta}_i$ is a function $\mathbf{Z}_{\text{an}^*(i)}$ for all i. Therefore, there exist node i and j such that $d^-(j,i) = \delta$ (notice that this implies that j is an ancestor of i). Thus, by Lemma 1, the GNN decoder has $N_h \geq \delta - 1$ hidden layers. The converse is true because Lemma 1 is a bi-conditional statement. □

Proposition 2 (Causal interventions). *VACA captures causal interventions if and only if the number of hidden layers in its decoder is greater than or equal to $\gamma - 1$, with γ being the length of the longest path between any two endogenous nodes in $\mathcal{G}$.*

A causal intervention involves to severe all the incoming edges to the intervened nodes. Thus, VACA can only capture causal interventions, if it can model all the causal paths, i.e., $P_{\text{GNN}}(j,i) = P(j,i)\ \forall i,j$. Otherwise, severing some paths will have no effect in the resulting intervention, as we prove next.

Proof. Consider a causal graph $\mathcal{G} := (\mathbf{X}, \mathbf{E})$ with length of the longest path between two nodes γ and a GNN decoder with N_h hidden layers. We assume that the GNN decoder models all the causal paths, i.e., $P_{\text{GNN}}(j,i) = P(j,i)\ \forall i,j$. By definition of γ, there exists at least one node i with an ancestor j such that $d^+(j,i) = \gamma$. Thus, by Lemma 1, the GNN decoder has $N_h \geq \gamma - 1$ hidden layers. The converse is true because Lemma 1 is a bi-conditional statement. □

Proposition 3 (Abduction). *The abduction step of an observed sample $\mathbf{x} = \{x_1, \ldots, x_d\}$ in VACA satisfies that for all i the posterior of Z_i is independent on the subset $\{x_j\}_{j \notin \text{pa}^*(i)} \subseteq \mathbf{x}$, if and only if the encoder GNN has no hidden layers.*

Proof. Consider a causal graph $\mathcal{G} := (\mathbf{X}, \mathbf{E})$ and a GNN encoder with N_h hidden layers. We assume the posterior of Z_i is independent on the subset $\{x_j\}_{j \notin \text{pa}^*(i)} \subseteq \mathbf{x}$, i.e., the parameters $\boldsymbol{\eta}_i^{enc}$ (the output of the GNN) is a function of $\{x_j\}_{j \in \text{pa}^*(i)}$. Then, the GNN only models paths $p(j,i)$ such that $d^+(j,i) = 1$. It follows, by Lemma 1, that the number of hidden layers of the encoder GNN is $N_h = 0$. The converse is true because Lemma 1 is a bi-conditional statement. □

C VACA implementation details

In this section, we extend Section 4 and provide further details about the implementation of VACA for complex real-world datasets and causal graphs.

Heterogeneous endogenous variables

As described in Appendix A, each layer l of a GNN uses the same parameters $\theta = \{\theta_m^l, \theta_u^l\}$ (corresponding to f^m and f^u) to update the features of every node, i.e., $h_i^l = f(\{h_i^{l-1}\} \cup \{h_j^{l-1} \mid j \in \mathcal{N}_i\}; \theta)$ with f_θ being reused for all i. Nonetheless, the structural equations of an SCM define a unique function f_i for each node (see **Property 1**). To mimic this behavior, we will rely on *port numbering*. In particular, for a given causal graph $\mathcal{G}$, we uniquely identify

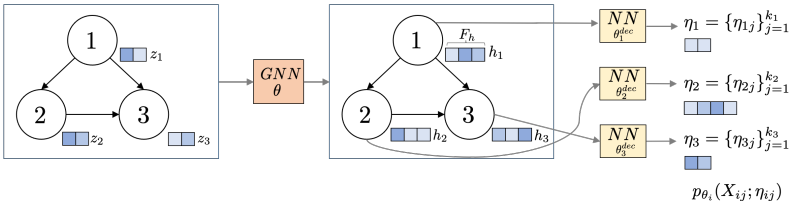


Figure 4: Heterogeneous VACA decoder architecture.

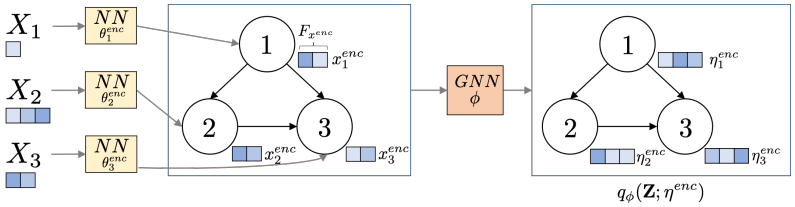


Figure 5: Heterogeneous VACA encoder architecture.

each node with an index i and each edge with the pair of indexes of the nodes it connects. Then, we define a *disjoint GNN layer* by the following characteristics:

- The node indexes define unique update functions f_i^u for each node, with parameters θ_{ui}.
- The edge indexes define unique message functions f_{ij}^m for each edge, with parameters θ_{mij}.

Consequently, parameters are not shared among nodes and we can mimic the diversity of the structural equations of an SCM and model heterogeneous endogenous variables. In our GitHub repository[2], we present a PyTorch Geometric implementation of the *disjoint GNN layer*.

Heterogeneous causal nodes

Assume an SCM with d endogenous variables. As described in Section 4, is it possible to model an endogenous variable $\mathbf{X}_i$ of the SCM as a heterogeneous node, i.e., $\mathbf{X}_i = \{X_{i1}, \dots, X_{ik_i}\}$, where k_i is the number of random variables in node i. In this section we describe the implications this has on the design of the encoder and decoder of VACA.

Implications for the decoder. Given the heterogeneous nature of the nodes, the likelihood of VACA factorizes as follows

$$p_\theta(\mathbf{X} \mid \mathbf{Z}) = \prod_{i=1}^{d} p_{\theta_i}(\mathbf{X}_i; \boldsymbol{\eta}_i) = \prod_{i=1}^{d} \prod_{j=1}^{k_i} p_{\theta_i}(X_{ij}; \boldsymbol{\eta}_{ij})$$

$$\text{where } \boldsymbol{\eta}_i = \boldsymbol{\eta}_i(\mathbf{Z}_{\text{an}^*(i)}) \text{ and } \boldsymbol{\eta}_{ij} = \boldsymbol{\eta}_{ij}(\mathbf{Z}_{\text{an}^*(i)}).$$

Note, each $p(X_{ij}; \boldsymbol{\eta}_{ij})$ can be model with a different distribution, e.g., Gaussian or categorical. This means that the likelihood parameters $\boldsymbol{\eta}_{ij}$ for each random variable X_{ij} may be different depending on its distribution type. However, the decoder GNN transforms the latent features into different features $\mathbf{H} \in \mathbb{R}^{d \times F_h}$, where the feature vector of each node i has the same dimensionality F_h. As a consequence, $\mathbf{H}$ cannot model the diversity in the likelihood parameters $\boldsymbol{\eta}_i$. To overcome such limitation, we add at the output of the GNN decoder a neural network (NN) per node i with parameters θ_i^{dec}. Such a NN transforms h_i, i.e. the output features of node i, into the set of likelihood parameters of each node $\boldsymbol{\eta}_i = \{\boldsymbol{\eta}_{ij}\}_{j=1}^{k_i}$, such that the likelihood parameters of each random variable $\boldsymbol{\eta}_{ij}$ satisfy the constraints of the corresponding likelihood $p(X_{ij}; \boldsymbol{\eta}_{ij})$ (e.g., non-negativity of variance for a Gaussian distribution). See Figure 4 for an illustration.

Implications for the encoder. Due to the heterogeneous nature of nodes, each endogenous variable $\mathbf{X}_i = \{X_{i1}, \dots, X_{ik_i}\}$, can have a different number of random variables k_i and thus the node i corresponding to it in the GNN will have features of different dimensions. However, as described in Appendix A, a GNN takes as input in general a matrix feature $\mathbf{X} \in \mathbb{R}^{d \times F_{x^{enc}}}$. This implies, the features of every node share the same dimensionality $F_{x^{enc}}$. To overcome this limitation, we include for each node i a neural network (NN) with parameters θ_i^{enc} that transforms the corresponding heterogeneous random variable $\mathbf{X}_i$ into a feature vector with the dimension $F_{x^{enc}}$. See Figure 5 for an illustration.

D Validating & understanding VACA

Here, we empirically validate the design conditions of VACA and provide an analysis of the effect of the latent space dimension $\dim \mathbf{z}$ on the performance.

Validating VACA design conditions. In a first step we empirically validate our design choices for the VACA encoder and decoder. We show how the number of hidden layers N_h in the decoder affects the quality of the estimation of the observational and interventional distribution. We do so for three SCMs, with different values of longest shortest directed path δ and longest directed path γ. Our observations in Table 3 match our expectations. The *collider* ($\delta = \gamma = 1$) does not need any hidden layer to provide accurate estimate

[2] https://github.com/XXXX

N_h	*collider* ($\delta = 1, \gamma = 1$)		*triangle* ($\delta = 1, \gamma = 2$)		*chain* ($\delta = 2, \gamma = 2$)	
	MMD Obs. (%)	MMD Inter.(%)	MMD Obs.(%)	MMD Inter.(%)	MMD Obs.(%)	MMD Inter.(%)
0	1.86 ± 0.84	1.16 ± 0.86	21.76 ± 5.80	48.47 ± 12.57	9.69 ± 1.92	17.24 ± 2.82
1	1.31 ± 0.36	0.75 ± 0.17	9.17 ± 2.39	19.91 ± 4.75	6.44 ± 1.52	10.36 ± 2.67
2	1.32 ± 0.41	1.09 ± 1.08	7.19 ± 3.14	14.10 ± 6.45	4.35 ± 1.77	6.55 ± 1.64

Table 3: Evaluation of the observational and interventional distributions generated by VACA with different numbers of hidden layers N_h in the decoder. All metrics are multiplied by 100 (%).

	collider		*triangle*		*chain*	
$\dim \mathbf{z}$	NLIN	NADD	NLIN	NADD	NLIN	NADD
2	3785	3785	4542	4542	3785	3785
4	4445	4445	5334	5334	4445	4445
8	5765	5765	6918	6918	5765	5765
16	8405	8405	10086	10086	8405	8405
32	13685	13685	16422	16422	13685	13685

Table 4: Number of parameters of VACA for different $\dim \mathbf{z}$ and datasets

of both the observational and interventional distributions. In contrast, the *triangle* ($\delta = 1, \gamma = 2$) – in accordance with **Proposition 2** – needs at least one hidden layer to get a more accurate estimate of the interventional distribution. Finally, as stated by **Propositions 1** and **2**, the *chain* ($\delta = \gamma = 2$) requires at least one hidden layer to accurately approximate both the observational and interventional distributions.

Analysis of the latent space dimension. Here we present an analysis of the performance of VACA with respect to the dimension of the latent space $\dim \mathbf{z}$. Without loss of generality, we focus the analysis on the *collider*, *triangle* and *chain* graphs, and the NLIN and NADD structural equations. The results are depicted in Figure 6. All results are averaged over 10 different initializations. Regarding the observational distribution (top row), we observe that VACA overfits as we increase the $\dim \mathbf{z}$ for all SEMs and graphs under consideration. Specially, the performance degrades notoriously with $\dim \mathbf{z} = 32$. As shown in Table 4, the number of parameters of VACA increases linearly with $\dim \mathbf{z}$. As for the interventional distribution (middle row), we observe similar behavior: for large values of $\dim \mathbf{z}$ performance decreases and variance increases. For counterfactuals (bottom row), we observe a similar behavior but considerably less pronounced. In summary, we encourage practitioners to keep $\dim \mathbf{z}$ small to avoid overfitting and obtain better and more consistent performance.

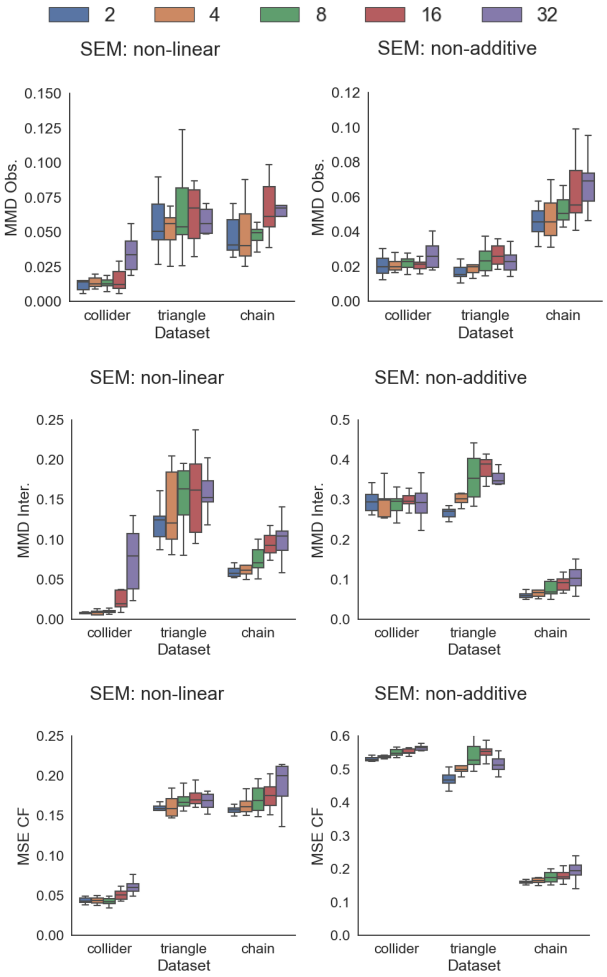


Figure 6: Performance of VACA varying the dimension of the latent space $\dim \boldsymbol{z} \in \{2, 4, 8, 16, 32\}$ for the non-linear (left column) and non-additive (right column) SEMs of the *collider*, *triangle* and *chain* graphs. We evaluate the quality of the observational (MMD Obs.) (top row), interventional (MMD Inter.) (middle row), and counterfactual (MSE CF) (bottom row) distributions.

E Experiments: setting, metrics and further results

This section provides a complete description of the experimental set-up, including the (semi-)synthetic datasets (Section E), training of VACA, MultiCVAE (Karimi et al. 2020) and CAREFL (Khemakhem et al. 2021) (Section E), metrics reported in the experiments (Section E), additional results (Section E), complexity of the algorithm (Section E), and computing infrastructure (Section E).

Datasets

The following (semi-)synthetic datasets are taken from or inspired by (Karimi et al. 2020). The distribution of exogenous variables $p(\mathbf{U})$ for *triangle*, *chain*and *collider* follows Table 5 with with MoG denoting a mixture of Gaussian distributions.

Collider. The *collider* is a synthetic dataset, which consists of 3 endogenous variables. The structural equations are shown in Table 6. Figure 7 illustrates the corresponding causal graph with $d = |\mathbf{X}| = 3$ nodes, diameter $\delta = 1$ and longest path $\gamma = 1$.

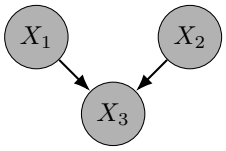


Figure 7: Causal graph for variables $\mathbf{X}$ of SCM *collider*.

Triangle. The *triangle* is a synthetic dataset, which consists of 3 endogenous variables. The structural equations are shown in Table 6. Figure 8 illustrates the corresponding causal graph with $d = |\mathbf{X}| = 3$ nodes, diameter $\delta = 1$ and longest path $\gamma = 2$.

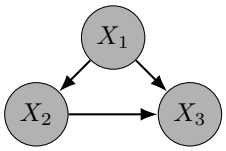
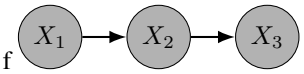


Figure 8: Causal graph for variables $\mathbf{X}$ of SCM *triangle*.

Chain. The *chain* is a synthetic dataset, which consists of 3 endogenous variables. The structural equations are shown in Table 6. Figure 9 illustrates the corresponding causal graph with $d = |\mathbf{X}| = 3$ nodes, diameter $\delta = 2$ and longest path $\gamma = 2$.



f

Figure 9: Causal graph for variables $\mathbf{X}$ of SCM *chain*.

M-graph. The *M-graph* is a synthetic dataset, which consists of 5 endogenous variables. Here, the distributions of exogenous variables follow $U_i \sim p(U_i) = \mathcal{N}(0, 1)\, \forall i \in 1 \dots 5$. The structural equations are shown in Table 7 and Figure 10 illustrates the corresponding causal graph with $d = |\mathbf{X}| = 5$ nodes, diameter $\delta = 1$ and longest path $\gamma = 1$.

Loan. The *loan* is a semi-synthetic dataset from (Karimi et al. 2020), which reflects a loan approval setting in the real-world inspired by German Credit dataset (Dua and Graff 2017a). It consists of 7 endogenous variables: *gender* G, *age* A, *education* E, *loan amount* L, *loan duration* D, *income* I

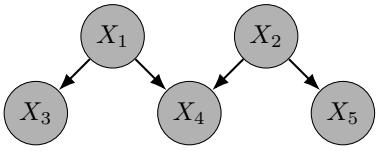


Figure 10: Causal graph for variables $\mathbf{X}$ of SCM *M-graph*.

and *savings* S with the following structural equations and distributions of exogenous variables:

$$
\begin{aligned}
f_G :& G = U_G \\
f_A :& A = -35 + U_A \\
f_E :& E = -0.5 + \left(1 + e^{+1-0.5G-\left(1+e^{-0.1A}\right)^{-1}-U_E}\right)^{-1} \\
f_L :& L = 1 + 0.01(A-5)(5-A) + G + U_L \\
f_D :& D = -1 + 0.1A + 2G + L + U_D \\
f_I :& I = -4 + 0.1(A+35) + 2G + GE + U_I \\
f_S :& S = -4 + 1.5\mathbb{I}_{\{I>0\}} I + U_S
\end{aligned}
$$

with $U_G \sim \mathrm{Bernoulli}(0.5)$, $U_A \sim \mathrm{Gamma}(10, 3.5)$, $U_E \sim \mathcal{N}(0, 0.25)$, $U_L \sim \mathcal{N}(0, 4)$, $U_D \sim \mathcal{N}(0, 9)$, $U_S \sim \mathcal{N}(0, 25)$, $U_I \sim \mathcal{N}(0, 4)$.

Note, the authors model variables w.r.t. their relative meaning in terms of deviation from the mean. See (Karimi et al. 2020) for further details. Figure 11 illustrates the corresponding causal graph with $d = |\mathbf{X}| = 7$ nodes, diameter $\delta = 2$ and longest path $\gamma = 3$.

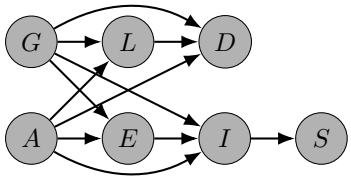


Figure 11: Causal graph for variables $\mathbf{X}$ of SCM *loan*.

Adult. We introduce the new semi-synthetic *adult* dataset which aims to reflect the relationship between the variables that affect the annual income of a person inspired by the real-world Adult dataset (Dua and Graff 2017b). Figure 13 shows the corresponding causal graph with $d = |\mathbf{X}| = 11$ nodes, diameter $\delta = 2$ and longest path $\gamma = 6$. We base the causal graph on (Chiappa 2019). Our dataset consists of the following 11 endogenous variables:

- *Race* R is an independent value.
- *Age* A is an independent value.
- *Native country* (N) is an independent value.
- *Sex* S is an independent value.
- *Education* level (E) depends on the sex, the native county and the race. We consider low, medium and high level of education.
- *Hours (worked) per week* H depends additionally on the race, the native country and the sex. We consider this continuous variable to be between 0 and 80 hours.

SCM	$p(U_1)$	$p(U_2)$	$p(U_3)$
LIN	$\text{MoG}(0.5\mathcal{N}(-2, 1.5) + 0.5\mathcal{N}(1.5, 1))$	$\mathcal{N}(0, 1)$	$\mathcal{N}(0, 1)$
NLIN	$\text{MoG}(0.5\mathcal{N}(-2, 1.5) + 0.5\mathcal{N}(1.5, 1))$	$\mathcal{N}(0, 0.1)$	$\mathcal{N}(0, 1)$
NADD	$\text{MoG}(0.5\mathcal{N}(-2.5, 1) + 0.5\mathcal{N}(2.5, 1))$	$\mathcal{N}(0, 0.25)$	$\mathcal{N}(0, 0.0625)$

Table 5: Distribution of exogenous variables $p(\mathbf{U})$ for SCM *triangle*, *chain*, *collider*.

	SCM	$\tilde{f}_1 := X_1$	$\tilde{f}_2 := X_2$	$\tilde{f}_3 := X_3$
collider	LIN	U_1	U_2	$0.05X_1 + 0.25X_2 + U_3$
	NLIN	U_1	U_2	$0.05X_1 + 0.25(X_2)^2 + U_3$
	NADD	U_1	U_2	$-1 + 0.1\,\text{sgn}(U_3)((X_1)^2 + (X_2)^2)U_3$
triangle	LIN	U_1	$-X_1 + U_2$	$X_1 + 0.25X_2 + U_3$
	NLIN	U_1	$-1 + \frac{3}{(1+\exp(-2X_1))} + U_2$	$X_1 + 0.25(X_2)^2 + U_3$
	NADD	U_1	$0.25\,\text{sgn}(U_2) * (X_1)^2(1 + (U_2)^2)$	$-1 + 0.1\,\text{sgn}(U_3)((X_1)^2 + (X_2)^2) + U_3$
chain	LIN	U_1	$-X_1 + U_2$	$0.25 * X_2 + U_3$
	NLIN	U_1	$-1 + \frac{3}{(1+\exp(-2X_1))} + U_2$	$0.25 * (X_2)^2 + U_3$
	NADD	U_1	$0.25\,\text{sgn}(U_2)(X_1)^2(1 + (U_2)^2)$	$-1 + 0.1\,\text{sgn}(U_3)((X_2)^2) + U_3$

Table 6: Structural equations $\tilde{\mathbf{F}}$ for different SCMs with $\mathbf{U} \sim p(\mathbf{U})$ in Table 5. Function $\text{sgn}(x)$ returns an element-wise indication of the sign of x.

SCM	$\tilde{f}_1 := X_1$	$\tilde{f}_2 := X_2$	$\tilde{f}_3 := X_3$	$\tilde{f}_4 := X_4$	$\tilde{f}_5 := X_5$
LIN	U_1	U_2	$X_1 + U_3$	$-X_2 + 0.5X_1 + U_4$	$-1.5X_2 + U_5$
NLIN	U_1	U_2	$X_1 + 0.5(X_1)^2 + U_3$	$-X_2 + 0.5(X_1)^2 + U_4$	$-1.5(X_2)^2 + U_5$
NADD	U_1	U_2	$X_1 * U_3$	$(-X_2 + 0.5 * (X_1)^2)U_4$	$(-1.5(X_2)^2)U_5$

Table 7: Structural Equations $\tilde{\mathbf{F}}$ for SCM *M-graph* with $U_i \sim p(U_i) = \mathcal{N}(0, 1)\ \forall i \in 1 \dots 5$.

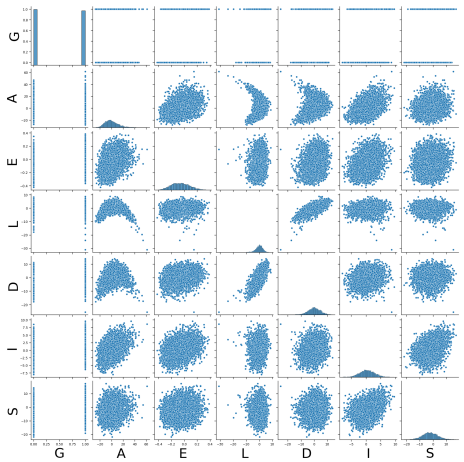


Figure 12: Histograms and scatter plots of pairwise feature relations for the *loan* dataset.

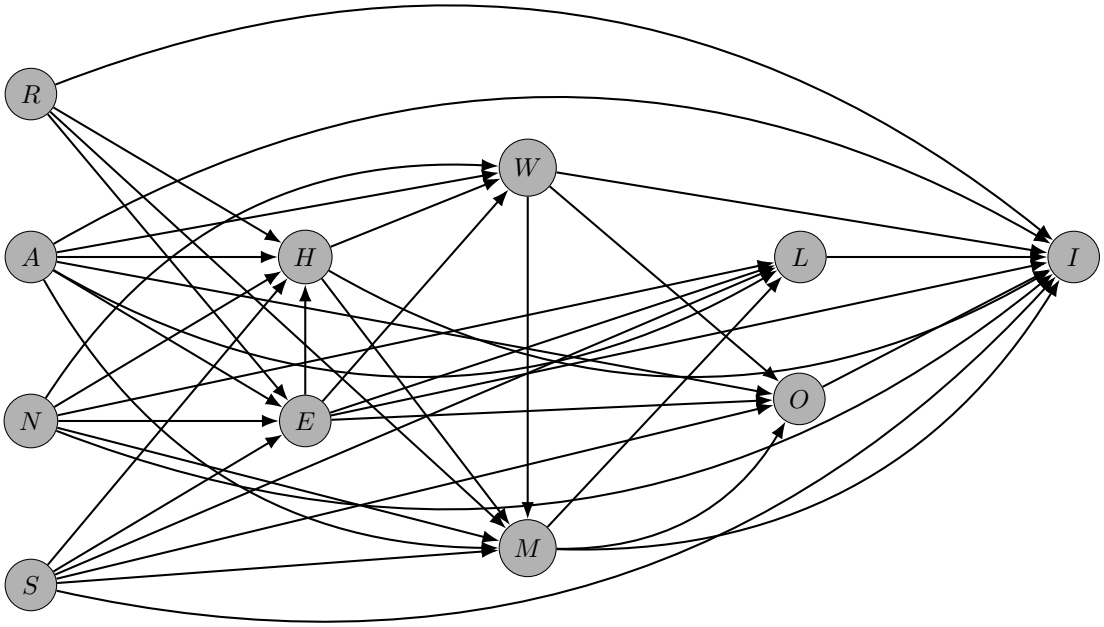


Figure 13: Causal graph for variables $\mathbf{X}$ of SCM *adult*.

- *Work status* W, i.e., being unemployed, self-employed, working for a company or in the public sector, is a discrete variable that depends on the number of hours worked per day, age, native country, and level of education.

- *Martial status* M, i.e., married never married or separated, depends on age, race, work status, hours per week, native country and sex. Individuals under a certain age are never married.

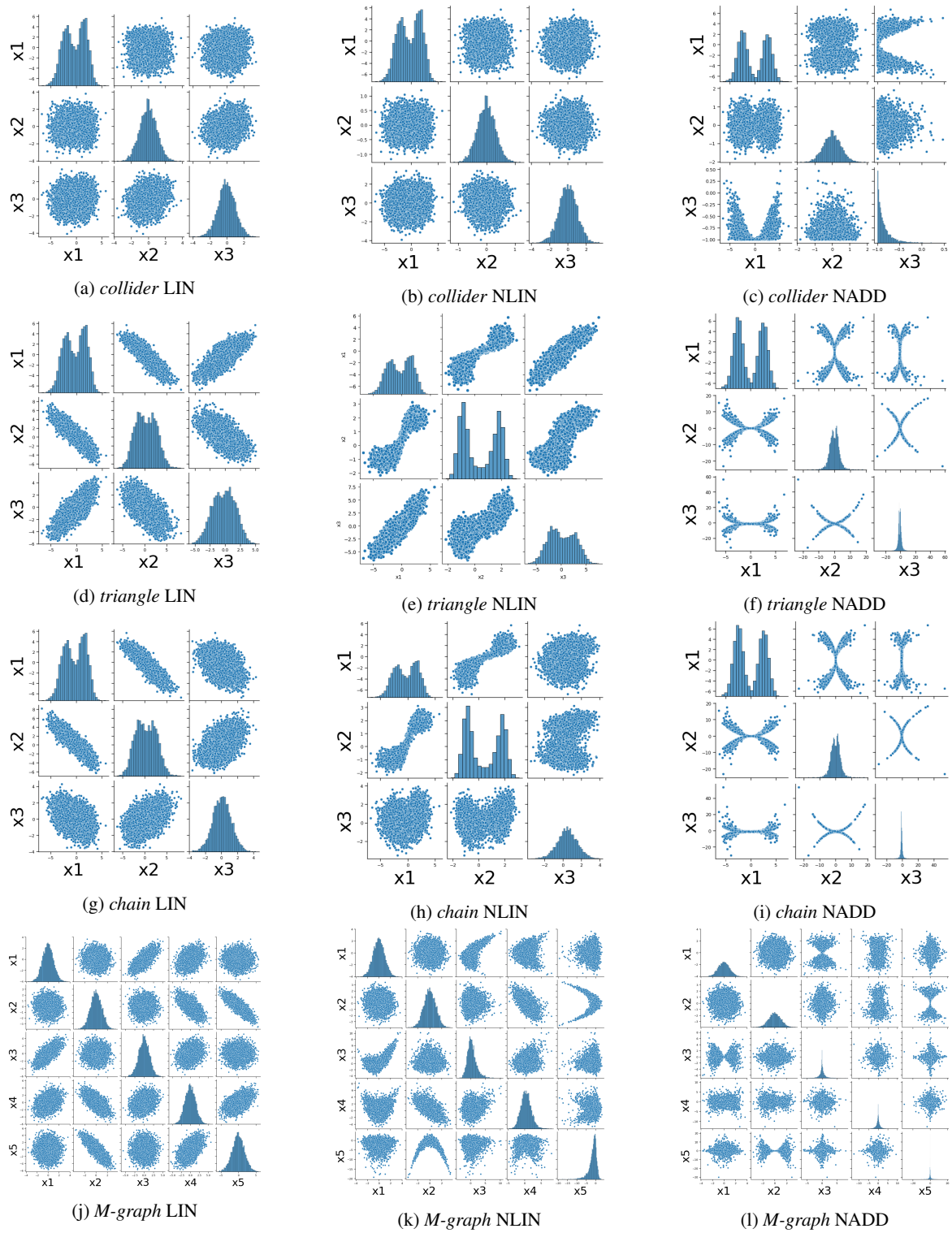
- *Occupation* sector O is a discrete variable with values technology , social sciences and medicine. It depends on the race, age, education, martial status and sex.

- *Relationship status* L is a discrete variable, with values wife, own-child, husband, not-in-family, unmarried. It depends on martial status, education, age, native country, and sex.

- *Income* I depends on race, age, education, occupation, work status, martial status, hours per week, relationship status, native country, and sex.

These aspects are reflected in the following structural equations :

$$
\begin{aligned}
&f_R : R = U_R \\
&f_A : A = U_A + 17 \\
&f_N : N = U_N \\
&f_S : S = U_S \\
&f_E : E = \exp\left(2\mathbb{I}_{\{R=0\}}) + \mathbb{I}_{\{R=1\}}\right) + \sigma(A-30)\big) \\
&\quad + (0.5\mathbb{I}_{\{S=0\}} + \mathbb{I}_{\{S=1\}})(2\mathbb{I}_{\{N=1\}} + 5\mathbb{I}_{\{N=2\}} \\
&\quad + \mathbb{I}_{\{N=3\}}) + U_E \\
&f_H : H = ((40\mathbb{I}_{\{N=0\}} + 36\mathbb{I}_{\{N=1\}} + 50\mathbb{I}_{\{N=2\}} \\
&\quad + 30\mathbb{I}_{\{N=3\}}) * \left(0.5\mathbb{I}_{\{R=0\}} + \mathbb{I}_{\{R=1\}} + 1.3\mathbb{I}_{\{R=2\}}\right) \\
&\quad + 2\exp\left(-(A-30)^2\right) + 5 \mid \tanh(E-2) \mid \\
&\quad + 2\mathbb{I}_{\{S=0\}} + U_H)\mathbb{I}_{\{A<70\}}
\end{aligned}
$$



(a) *collider* LIN (b) *collider* NLIN (c) *collider* NADD

(d) *triangle* LIN (e) *triangle* NLIN (f) *triangle* NADD

(g) *chain* LIN (h) *chain* NLIN (i) *chain* NADD

(j) *M-graph* LIN (k) *M-graph* NLIN (l) *M-graph* NADD

Figure 14: Histograms and scatter plots of pairwise feature relations for the *collider*, *triangle*, *chain* and *M-graph* datasets.

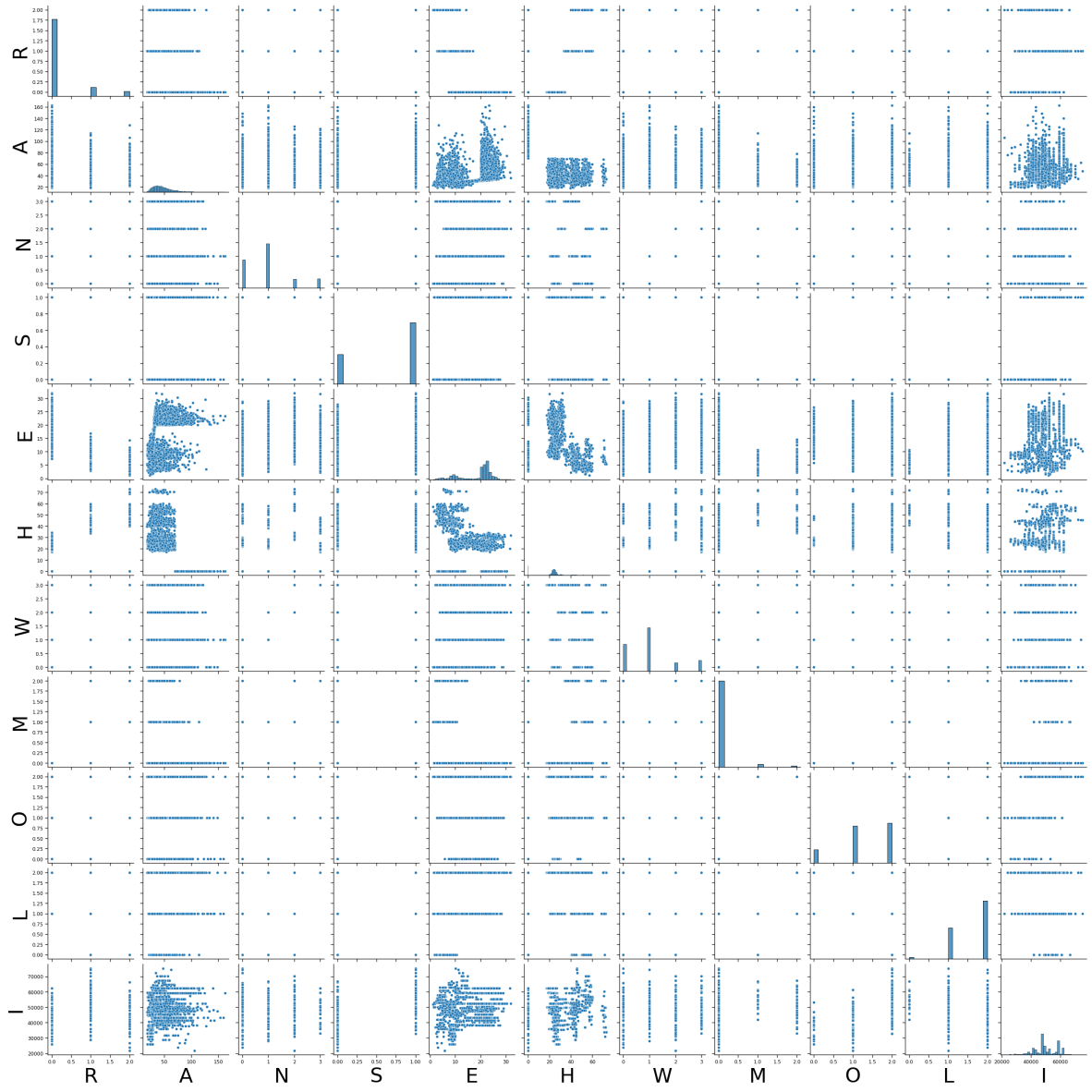


Figure 15: Histograms and scatter plots of pairwise feature relations for the *adult* dataset.

$$
\begin{aligned}
f_W &: W = w_2 \mathbb{I}_{\{w_2>=0\}}, \\
&\quad w_2 = w_1 \mathbb{I}_{\{w_1<=3\}} + 3\mathbb{I}_{\{w_1>3\}} \\
&\quad w_1 = \mathbb{I}_{\{5|\tanh(E-2)|+\sigma(H-30+U_W)>0.3\}} \\
&\quad + \mathbb{I}_{\{\sigma(H-30+U_W>0.3\}} \mathbb{I}_{\{(A+1.5U_W)>50\}} \\
&\quad - \mathbb{I}_{\{N=0\}} + \mathbb{I}_{\{N=2\}} + 3\mathbb{I}_{\{N=3\}}) \\
f_M &: M = \text{mode}(r_2, a_2, W, h_2, H, g_3) \\
&\quad r_2 = 2 * \mathbb{I}_{\{r_1=1\}} + \mathbb{I}_{\{r_1=2\}} \\
&\quad r_1 = \text{int}(R + 0.2U_M) \mathbb{I}_{\{R\in[0,2]\}} + 2\mathbb{I}_{\{R>2\}} \\
&\quad g_3 = \mathbb{I}_{\{g_2=0\}} + 2 * \mathbb{I}_{\{g_2=1\}} \\
&\quad g_2 = 0 * \mathbb{I}_{\{g_1<0\}} + \mathbb{I}_{\{g_1>1\}} + g_1 \mathbb{I}_{\{g_1\in[0,1]\}} \\
&\quad g_1 = \text{int}(G + 0.5U_M) \\
&\quad a_2 = 2\mathbb{I}_{\{a_1\in(20,40]\}} + \mathbb{I}_{\{a_1\in(40,50]\}} + 2\mathbb{I}_{\{a_1>=50\}} \\
&\quad a_1 = A + 2U_M \\
&\quad h_2 = h_1 \mathbb{I}_{\{h_1<=2\}} + 2\mathbb{I}_{\{h_1>2\}} \\
&\quad h_1 = 3\,\text{int}(\sigma(H - 30)) \\
f_L &: L = 0\mathbb{I}_{\{(M=1)\wedge(c<-1)\}} + 1\mathbb{I}_{\{(M=1)\wedge(c\geq-1)\}} \\
&\quad + 2\mathbb{I}_{\{(M\neq1)\wedge(c\geq-1)\}} + 1\mathbb{I}_{\{(M\neq1)\wedge(c<-1)\}} \\
&\quad c = c_n + c_e + 2\mathbb{I}_{\{A<20\}} - 2\mathbb{I}_{\{S=0\}} \\
&\quad c_n = U_O \mathbb{I}_{\{N=0\}} - U_O \mathbb{I}_{\{N=1\}} + 2U_O \mathbb{I}_{\{N=2\}} + 2\mathbb{I}_{\{N=3\}} \\
&\quad c_e = \sigma(E - 30) \\
f_O &: O = 0\mathbb{I}_{\{k<1\}} + 1\mathbb{I}_{\{1\leq k\leq4\}} + 2\mathbb{I}_{\{4<k\}} \\
&\quad k = R + k_a + k_e + W + 3M + 4S \\
&\quad k_a = 2e^{-(A+U_O-20)^2} \\
&\quad k_e = -\sigma(E * U_O - 30) \\
f_I &: I = U_I \\
&\quad + 10,000\mathbb{I}_{\{R>1.5\}} + 20,000\mathbb{I}_{\{R<1.5\}} \\
&\quad + 3,000\mathbb{I}_{\{21\leq A<30\}} + 8,000\mathbb{I}_{\{30\leq A\}} \\
&\quad + 5,000\mathbb{I}_{\{E<2\}} + 10,000\mathbb{I}_{\{2\leq E<10\}} + 30,000\mathbb{I}_{\{10\leq E\}} \\
&\quad + 5,000\mathbb{I}_{\{O=1\}} + 15,000\mathbb{I}_{\{O=2\}} \\
&\quad + 5,000\mathbb{I}_{\{W=0\}} + 7,000\mathbb{I}_{\{W=1\}} \\
&\quad + 1,000\mathbb{I}_{\{M=0\}} + 4,000\mathbb{I}_{\{M=1\}} - 2,000\mathbb{I}_{\{M=2\}} \\
&\quad + 15,000\mathbb{I}_{\{H>45\}} + 10,000\mathbb{I}_{\{N\geq2\}} \\
&\quad + 4,000\mathbb{I}_{\{S=1\}} + 3,000\mathbb{I}_{\{R\leq1\}}
\end{aligned}
$$

Training and cross-validation

This section details the hyperparameter cross-validation of VACA, MultiCVAE (Karimi et al. 2020) and CAREFL (Khemakhem et al. 2021) for the experiments in Tables 1 and Table 12. Across experiments and models we generate synthetic datasets consisting of 5000 training samples, 2500 test samples and 2500 validation samples and we use a batch size of 1000. Al the models have been cross-validated using a similar computational budget, i.e. the number of combinations of hyperparameters cross-validated is the same for all the models (see Table 8). Additionally, we run each configuration over 10 different random initializations. Regarding the number of iterations, we assumed a large enough number of epochs for training loss to converge, and implemented early stopping to prevent overfitting the training data. The best configuration has been chosen in terms of the best observational MMD.

SCM	MultiCVAE	CAREFL	VACA
chain	72	72	72
collider	72	72	72
triangle	72	72	72
M-graph	72	72	72
loan	40	24	24

Table 8: Number of combinations of hyperparameters cross-validated for each of the models and datasets. Note that for each combination we run 10 different seeds.

VACA. We optimize the ELBO (Kingma and Welling 2014) and use the IWAE (Burda, Grosse, and Salakhutdinov 2016) with $K = 100$ as the objective metric for the early stopping procedure. We use the Rectified Linear Unit (ReLU) as activation function. We trained with a learning rate $\eta = 0.005$ for a maximum of 500 epochs, or alternatively until the objective metric does not improve in 50 epochs. Also, we regularize the training of VACA using a novel *parents dropout:* randomly removing all incoming edges to the nodes with probability $p \in [0, 1)$. In our experiments, we observe that adding this regularization improves overall performance. We cross-validated the parents dropout rate with values $\{0.1, 0.2\}$, the number of hidden layers of the decoder with values $\{0, 1, 2, 3, 4, 5\}$ (the specific values depend on the diameter of the graph) with 16 neurons each, and weather to use or not residual connections. The best models are reported in Table 9. We use a latent variable dimension of 4 and a Gaussian likelihood with a small variance $\sigma^2 = \lambda_{KLD}/2$ with $\lambda_{KLD} = 0.05$.

MultiCVAE. Karimi et al. (2020) propose to train a conditional variational autoencoder (CVAE) for each endogenous variable that is not a root node in the causal graph, we refer to as MultiCVAE. Different from (Karimi et al. 2020), our implementation also models non-root nodes as CVAEs, since our goal is to model the joint distribution, while (Karimi et al. 2020) target interventional and counterfactual distributions for algorithmic recourse only. Additionally, we perform the necessary modifications for training on normalized data.

We cross-validated the number and size of hidden layers of the decoder, the dropout rate, and the dimension of the latent space. The best models (according to the observational MMD) are reported in Table 10. As with VACA, we assume $\lambda_{KLD} = 0.05$ for all SCMs and CVAEs.

CAREFL. Khemakhem et al. (2021) propose CAREFL, an autoregressive causal flows model for causal discovery, which also allows to answer interventional and counterfactual queries. The authors rely on real-valued non-volume

SCM		Encoder Arch.	Decoder Arch.	Parents Dropout	Residual
chain	LIN	$1 \times 16 \times 4$	$4 \times 16 \times 1$	0.1	1
	NLIN	$1 \times 16 \times 4$	$4 \times 16 \times 16 \times 1$	0.1	0
	NADD	$1 \times 16 \times 4$	$4 \times 64 \times 1$	0.1	1
collider	LIN	$1 \times 16 \times 4$	$4 \times 16 \times 16 \times 1$	0.2	1
	NLIN	$1 \times 16 \times 4$	$4 \times 16 \times 1$	0.1	0
	NADD	$1 \times 16 \times 4$	$4 \times 16 \times 16 \times 1$	0.1	1
triangle	LIN	$1 \times 16 \times 4$	$4 \times 16 \times 16 \times 1$	0.1	0
	NLIN	$1 \times 16 \times 4$	$4 \times 16 \times 16 \times 1$	0.2	0
	NADD	$1 \times 16 \times 4$	$4 \times 16 \times 16 \times 1$	0.1	0
M-graph	LIN	$1 \times 16 \times 4$	$4 \times 16 \times 1$	0.1	0
	NLIN	$1 \times 16 \times 4$	$4 \times 16 \times 16 \times 1$	0.1	0
	NADD	$1 \times 16 \times 4$	$4 \times 16 \times 16 \times 1$	0.1	0
loan	-	$1 \times 16 \times 16 \times 4$	$4 \times 16 \times 16 \times 16 \times 1$	0.1 2	0
adult	-	$1 \times 16 \times 16 \times 4$	$4 \times 8 \times 8 \times 8 \times 8 \times 1$	0.2	1

Table 9: Hyperparemeter selection for our VACA training for the SCMs on the synthetic datasets *triangle*, *collider*, *chain* and *M-graph* and on the semi-synthetic dataset *loan*. Note, the encoder architecture refers to the layers in function f^m, while the decoder architecture refers to the different GNN layers.

SCM		λ_{KLD}	Encoder Arch.	Decoder Arch.	Dropout rate
chain	LIN	0.05	$1 \times 16 \times 4$	$4 \times 128 \times 1$	0.0
	NLIN	0.05	$1 \times 16 \times 1$	$1 \times 128 \times 1$	0.0
	NADD	0.05	$1 \times 16 \times 4$	$4 \times 128 \times 1$	0.0
collider	LIN	0.05	$1 \times 16 \times 4$	4×1	0.0
	NLIN	0.05	$1 \times 16 \times 4$	4×1	0.0
	NADD	0.05	$1 \times 16 \times 4$	4×1	0.0
triangle	LIN	0.05	$1 \times 16 \times 4$	$4 \times 128 \times 1$	0.2
	NLIN	0.05	$1 \times 16 \times 1$	$1 \times 128 \times 1$	0.1
	NADD	0.05	$1 \times 16 \times 4$	$4 \times 128 \times 1$	0.2
M-graph	LIN	0.05	$1 \times 16 \times 4$	4×1	0.2
	NLIN	0.05	$1 \times 16 \times 4$	4×1	0.1
	NADD	0.05	$1 \times 16 \times 4$	$4 \times 32 \times 32 \times 1$	0.2
loan	-	0.05	$1 \times 16 \times 16 \times 1$	$1 \times 64 \times 64 \times 1$	0.0
adult	-	0.05	$1 \times 16 \times 16 \times 2$	$2 \times 16 \times 1$	0.0

Table 10: Hyperparemeter selection for MultiCVAE (Karimi et al. 2020) training for the SCMs on the synthetic datasets with three nodes (i.e., *triangle*, *collider* and *chain*), *M-graph* and for the semi-synthetic dataset *loan*.

SCM		Flows	Hidden Units
chain	LIN	4	5
	NLIN	3	64
	NADD	5	5
collider	LIN	2	96
	NLIN	2	64
	NADD	3	16
triangle	LIN	4	64
	NLIN	4	5
	NADD	3	96
M-graph	LIN	4	64
	NLIN	4	96
	NADD	5	96
loan	-	4	64
adult	-	2	96

Table 11: Hyperparemeter selection for CAREFL (Khemakhem et al. 2021) training for different SCMs.

preserving (real NVP) transformations, since they mainly focus on the multivariate bi-variate case. As this flow architecture is not suited for general graphs, we use their framework with Neural Spline Autoregressive Flows (Durkan et al. 2019). We have cross validated the number of flows $\{2, 3, 4, 5, 6\}$ and the number of hidden units of the neural networks $\{5, 10, 16, 32, 64, 96\}$. The final configuration– i.e., the configuration with the lowest observational MMD– is displayed in Table 11.

Performance metrics

In the following we describe the metrics used to evaluate the performance of VACA in Section 5. In all experiments we use (semi-)synthetic datasets with access to samples from the ground truth distribution $\{\mathbf{x}_i\}_{i=0}^n \sim P$ as well as from the estimated distribution $\{\hat{\mathbf{x}}_i\}_{i=0}^n \sim Q$.

For the interventional and counterfactual distribution, we perform a set of interventions $\mathcal{I} = \{do(\mathbf{X}_{\mathcal{I}_j} = \alpha_j)\}_j$, where $\mathcal{I}_j \in [d]$ and $\alpha_j \in \{-1.0, -0.5, 0.0, 0.5, 1.0\} \times \sigma_{\mathcal{I}_j}$ with $\sigma_{\mathcal{I}_j}$ as the empirical standard deviation of the intervened variable $\mathbf{X}_{\mathcal{I}_j}$ prior to intervention (i.e., in the observational distribution). Note that we only intervene on one variable at a time. For each intervention in $\mathcal{I}$, we are interested in the estimated distribution of variables causally affected by the intervention $\{\mathbf{X}_i | i \in des(\mathcal{I}_j)\}$, i.e., the set of descendants of the variable intervened. Note that $des(\mathcal{I}_j)$ refers to the set of indexes of the descendants. It follows that we do not intervene on leaf nodes.

Mean Maximum Discrepancy (MMD). The Mean Maximum Discrepancy (MMD) (Gretton et al. 2012) is a kernel-based distance-measure between two distributions P and Q on the basis of samples from both distributions. The smaller the MMD, the more likely it is that the sets of samples are drawn from the same distributions, i.e. the better distributions match.

Without access to underlying distribution, we can compute an unbiased empirical squared MMD estimate using a kernel function k as(Gretton et al. 2012):

$$\widehat{\mathrm{MMD}}^2(\mathbf{X}, \hat{\mathbf{X}}) = \frac{1}{n(n-1)} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} k(\mathbf{x}_i, \mathbf{x}_j) + \sum_{i=1}^{n} \sum_{j=1}^{n} k(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j) - 2 \sum_{i=1}^{n} \sum_{j=1}^{n} k(\mathbf{x}_i, \hat{\mathbf{x}}_j) \right). \tag{10}$$

In our implementation we use as kernel a mixture of RBF (Gaussian) kernels with different bandwidths and sample size $n = 1000$.

Estimation squared error for the mean (MeanE). For the interventional distribution, we compute the estimation squared error for the mean (MeanE) as the average (across interventions) of the squared difference between the empirical means of the true and estimated interventional distributions (for the descendants of the intervened variables):

$$\text{MeanE} = \frac{1}{|\mathcal{I}|} \sum_{\mathcal{I}_j \in \mathcal{I}} \frac{1}{|des(\mathcal{I}_j)|} \times \sum_{i \in des(\mathcal{I}_j)} \left(E[X_i^{\mathcal{I}_j}] - E[\hat{X}_i^{\mathcal{I}_j}] \right)^2 \tag{11}$$

Estimation squared error for the standard deviation (StdE). For the interventional distribution, we compute the estimation squared error for the standard deviation (StdE) as the average (across interventions) of the squared difference between the empirical standard deviation of the true $\tilde{\sigma}(X_i^{\mathcal{I}_j})$ and estimated $\tilde{\sigma}(\hat{X}_i^{\mathcal{I}_j})$ interventional distributions (for the descendants of the intervened variables):

$$\text{StdE} = \frac{1}{|\mathcal{I}|} \sum_{\mathcal{I}_j \in \mathcal{I}} \frac{1}{|des(\mathcal{I}_j)|} \times \sum_{i \in des(\mathcal{I}_j)} \left(\tilde{\sigma}(X_i^{\mathcal{I}_j}) - \tilde{\sigma}(\hat{X}_i^{\mathcal{I}_j}) \right)^2 \tag{12}$$

Mean squared error (MSE). For the counterfactual distribution, we compute the mean squared error (MSE) as the average (across interventions) of the pairwise squared difference between true and estimated counterfactual values for the descendants of the intervened variable. More in detail, let us define the random variable $T^{\mathcal{I}_j}$ as the *Frobenius norm* of the difference between true $\mathbf{x}^{CF}_{des(\mathcal{I}_j)}$ and estimated $\hat{\mathbf{x}}^{CF}_{des(\mathcal{I}_j)}$ counterfactual values for the descendants of the intervened variable, i.e.,

$$T^{\mathcal{I}_j} = ||\mathbf{x}^{CF}_{des(\mathcal{I}_j)} - \hat{\mathbf{x}}^{CF}_{des(\mathcal{I}_j)}||_2^2, \tag{13}$$

Thus, we can compute the counterfactual MSE as:

$$\text{MSE} = \frac{1}{|\mathcal{I}|} \sum_{\mathcal{I}_j \in \mathcal{I}} \frac{1}{|des(\mathcal{I}_j)|} E\left[T^{\mathcal{I}_j}\right] \tag{14}$$

Standard deviation of the squared error (SSE). Similarly, we can compute the average (across interventions) of the standard deviation of the counterfactual squared error as:

$$\text{SSE} = \frac{1}{|\mathcal{I}|} \sum_{\mathcal{I}_j \in \mathcal{I}} \frac{1}{|des(\mathcal{I}_j)|} \sum_{i \in des(\mathcal{I})} \tilde{\sigma}\left(T^{\mathcal{I}_j}\right), \qquad (15)$$

where $\tilde{\sigma}\left(T^{\mathcal{I}_j}\right)$ denotes the empirical standard deviation of $T^{\mathcal{I}_j}$.

Additional results

In the following we present additional results that empirically show the potential of VACA to model interventional and counterfactual queries. In particular, we report the results for the *triangle*, *M-graph*, and *chain* graphs. We remark that the following results are consistent with the ones reported in the main manuscript for the *collider*, *loan* and *adult*.

Results for interventional distributions. Table 12 (middle columns) reports the MMD, MeanE, and StdE for the interventional distribution. In accordance with the results shown in the main manuscript, we can observe that i) VACA consistently outperforms other methods in terms of MMD; ii) the three methods provide comparable results in capturing the mean of the interventional distribution (MeanE); and iii) CAREFL and MultiCVAE often fail to capture the standard deviation of the interventional distribution (StdE), while VACA provides a more accurate estimate of the overall interventional distribution (as can be easily seen in the MMD).

Results for the counterfactuals. Table 12 also reports the results for the counterfactual distribution, in terms of MSE and SSE. As reported in the main text, we observe that CAREFL provides more accurate estimates than VACA and MultiCVAE in terms of MSE, which may be explained by the fact that CAREFL performs exact inference as opposed to the approximated inference of the other two approaches. However, CAREFL presents high variance in its results (see SSE). In contrast, VACA leads to regularly lower values of SSE, which suggests more consistent counterfactual estimations across factual samples and interventions.

Complexity analysis

In this section we compare the amount of time that it takes VACA and the two baselines to converge during training. Figure 16 shows the time (in minutes) it takes to train the models for the configurations of hyperparameters cross-validated (see Section E). Note we also show the results averaged over 10 different initializations. We can extract three main points from Figure 16. Firstly, we observe that most of the experiments take less than 60 minutes to converge, only some experiments with the *loan* graph take longer. Secondly, MultiCVAE takes longer to converge than CAREFL and VACA on average. This can be explained by the fact that nodes of the graph are trained sequentially. Thirdly, we observe that CAREFL and VACA take similar amount of time to converge on average. We remark that experiments were run in a computer cluster, whose performance depends on its congestion, i.e. the number of people using the cluster and the amount of experiments. Thus, it is possible that part of the variance in the times is due to different congestion situations in the cluster.

Figure 17 shows a box-plot of the number of parameters of the configuration of hyperparameters cross-validated for the models and the datasets under study. Firstly, we observe that the configurations chosen for MultiCVAE contain less number of parameters than the configurations of the other two methods. However, increasing the number of parameters leads to an increase in training time. MultiCVAE already has the longest training time compared to the other methods, so the comparison is unfair in terms of time complexity. Regarding CAREFL and VACA, we observe that the number of parameters cross-validated overlaps in all the cases.

Computing infrastructure

All the experiments conducted in this work were executed in the same computer cluster based on the Linux OS. Each experiment was assigned to 2 CPUs (they could be assigned only to 1 CPU but the more CPUs, the faster the data-loading process is) and 8GB of memory. For information about the required software packages please refer to the offical GitHub repository of VACA https://github.com/XXXX/XXXX.

F Further details on the counterfactual fairness use-case

In this section we provide further details on dataset, training, metrics and additional results for the use-case of counterfactual fairness in Section 6.

German Credit Dataset

The German Credit dataset from the UCI repository (Dua and Graff 2017a) contains 20 attributes from 1000 loan applicants. We rely on the causal model proposed by in (Chiappa 2019) for the following subset of features as exogenous variables $\mathbf{X}$ (see Figure 18): sensitive feature $S = \{sex\}$, and non-sensitive features $C = \{age\}$, $R = \{\textit{credit amount}, \textit{repayment history}\}$ and $H = \{\textit{checking account}, \textit{savings}, \textit{housing}\}$. The causal graph in Figure 18 has a diameter $\delta = 1$ and longest path $\gamma = 1$. The goal of a classifier h is to predict $Y = \{creditrisk\}$ from $\mathbf{X}$. We load and pre-process the data using the aif360 library such that the dataset contains binary outcome variable Y (0-bad, 1-good) and a binary sensitive attribute S (0-female, 1-male). Note that the dataset contains 700 labels $Y = 1$ and 300 labels $Y = 0$, i.e., it is imbalanced. It also contains 690 males $S = 1$ and 310 females $Y = 0$. Note also that the causal model contains heterogeneous causal nodes (R and S), as addressed in Section 4. For example, (Chiappa 2019) assume that the relationship between *credit amount* and *repayment history* is unknown, or that it may be affected by hidden confounders. This leads to an undirected path between the random variables and they are grouped together in one multidimensional causal node R. This applies similarly to node S.

SCM		Model	Obs. MMD	Interventional MMD	Interventional MeanE	Interventional StdE	Counterfactuals MSE	Counterfactuals SSE	Num. params
collider	LIN	MultiCVAE	30.37±8.16	44.70±12.25	13.29±4.78	46.56±2.40	87.41±3.64	65.15±2.83	553
		CAREFL	9.27±1.49	4.86±0.45	0.35±0.08	81.89±1.78	8.11±0.58	7.83±0.55	6420
		VACA	1.50±0.67	1.57±0.41	0.75±0.31	41.99±0.30	9.86±0.74	7.06±0.38	5600
	NLIN	MultiCVAE	28.03±9.12	41.60±12.62	10.49±4.12	46.48±2.43	82.32±2.61	62.05±1.87	553
		CAREFL	10.38±2.00	4.69±0.38	0.19±0.07	80.68±2.08	6.93±0.40	7.15±0.64	4308
		VACA	0.95±0.27	0.97±0.23	0.26±0.12	42.20±0.24	5.01±0.73	4.08±0.54	1805
	NADD	MultiCVAE	29.72±8.90	117.67±46.20	26.23±11.09	39.75±1.34	73.93±10.44	51.44±6.06	553
		CAREFL	9.79±1.29	9.13±1.45	1.58±0.94	102.13±3.16	9.92±0.51	32.66±0.86	1710
		VACA	1.71±0.48	29.87±1.94	10.71±1.05	50.25±0.87	31.99±0.84	39.10±0.67	5600
triangle	LIN	MultiCVAE	33.12±3.89	157.50±15.08	12.10±2.03	44.66±2.57	65.95±3.05	42.39±1.75	3243
		CAREFL	13.50±1.83	8.88±0.54	0.62±0.25	80.11±2.44	7.05±1.09	8.91±1.43	8616
		VACA	3.64±1.64	13.82±1.87	2.90±0.49	25.82±0.33	23.52±1.83	16.02±1.32	5334
	NLIN	MultiCVAE	46.65±9.05	218.23±30.38	17.50±4.85	27.55±1.98	52.34±4.50	32.45±2.22	1785
		CAREFL	13.55±2.25	19.01±1.05	1.61±0.44	103.03±2.26	9.38±4.21	10.38±4.20	828
		VACA	6.04±1.87	9.53±3.60	1.44±0.59	17.93±0.19	13.24±1.93	8.26±1.29	5334
	NADD	MultiCVAE	16.99±7.22	133.83±16.96	2.59±3.38	18.25±0.83	42.43±0.87	20.55±1.15	3243
		CAREFL	13.58±1.69	74.51±14.18	1.31±0.84	148.03±6.96	11.62±1.50	53.71±0.98	9630
		VACA	1.72±0.67	30.10±4.41	0.19±0.14	26.64±1.30	22.50±2.61	41.16±3.91	5334
M-graph	LIN	MultiCVAE	38.60±4.06	80.89±20.73	25.82±6.26	17.73±3.39	54.72±4.71	27.47±2.28	933
		CAREFL	19.55±3.48	15.38±0.75	0.77±0.23	68.32±1.35	6.94±0.23	9.97±0.15	18200
		VACA	1.76±0.34	4.60±0.81	1.86±0.23	11.79±0.11	12.83±0.69	9.10±0.51	3249
	NLIN	MultiCVAE	37.95±6.21	84.79±18.15	24.97±9.10	16.66±3.28	51.30±5.24	25.90±1.23	933
		CAREFL	20.72±2.78	18.63±1.36	2.57±0.69	74.41±1.02	9.43±0.34	24.94±0.31	27160
		VACA	1.95±0.40	6.78±0.95	2.27±0.41	12.55±0.43	16.26±0.95	17.65±1.12	8001
	NADD	MultiCVAE	19.97±5.03	80.25±14.12	0.92±0.60	32.94±1.77	38.28±1.00	32.61±1.11	7277
		CAREFL	19.36±1.59	18.37±0.78	0.28±0.09	79.71±2.30	32.66±0.31	49.78±0.19	33950
		VACA	3.06±0.84	17.80±1.78	0.09±0.05	21.22±0.56	41.06±0.31	48.39±0.35	8001
chain	LIN	MultiCVAE	29.38±2.96	131.24±12.04	6.75±1.66	38.96±2.17	59.15±1.61	41.37±1.07	3099
		CAREFL	11.70±1.46	12.00±1.23	0.98±0.38	81.15±2.55	9.90±1.42	11.72±1.80	828
		VACA	4.47±0.72	5.73±1.11	3.60±0.62	22.91±0.23	23.09±1.18	15.77±0.77	5648
	NLIN	MultiCVAE	30.33±4.53	159.00±11.81	8.68±4.51	43.49±2.91	60.34±1.87	40.88±1.08	1641
		CAREFL	12.35±2.03	11.08±1.02	1.04±0.43	79.06±2.16	10.56±0.74	12.05±1.17	6462
		VACA	4.66±1.06	6.17±1.24	3.69±0.46	22.41±0.45	23.17±2.04	15.69±1.25	4445
	NADD	MultiCVAE	29.65±2.89	132.74±11.58	7.10±1.75	40.59±2.23	59.21±1.66	41.26±1.08	3099
		CAREFL	11.24±1.41	10.86±1.04	1.15±0.32	80.26±2.52	10.18±2.11	12.92±2.86	1035
		VACA	4.67±0.91	6.09±1.16	3.98±0.72	23.14±0.20	23.58±1.33	15.96±0.81	5648
loan	-	MultiCVAE	90.38±11.31	213.65±5.38	12.24±1.33	65.78±1.13	40.98±0.35	15.12±0.16	33717
		CAREFL	22.10±1.64	27.38±4.07	6.74±4.25	50.13±2.47	11.15±2.57	6.59±0.38	2880
		VACA	2.22±0.25	6.87±0.66	4.35±0.35	3.83±0.08	10.30±0.40	6.41±0.11	30402
adult		MultiCVAE	140.15±6.37	155.52±5.93	12.18±2.36	63.52±4.05	39.96±0.36	16.37±0.65	6549
		CAREFL	31.31±1.58	34.31±5.77	12.54±3.17	41.26±3.44	1.23±0.17	3.55±0.90	127420
		VACA	4.51±0.45	12.68±1.95	1.65±0.23	3.37±0.09	5.33±0.27	5.67±0.20	63432

Table 12: Performance of different methods at estimating the observational, interventional and counterfactual of different SCMs. All metrics are multiplied by 100.

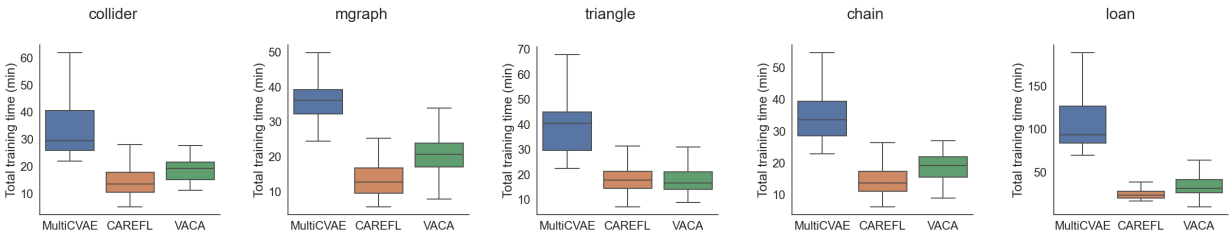


Figure 16: Training time (in minutes) for MultiCVAE, CAREFL and VACA for the different graphs.

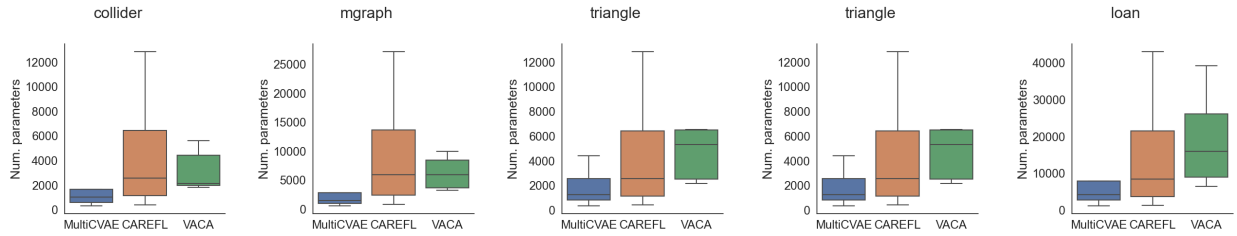


Figure 17: Number of parameters of MultiCVAE, CAREFL and VACA for the different graphs.

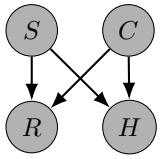


Figure 18: Causal graph for variables $\mathbf{X}$ of the German Credit dataset (Chiappa 2019)

Training

In this section we provide further information on training VACA on the German Credit dataset (Dua and Graff 2017a) and detail the different classifiers in Section 6. We use a 80% training, 10% validation, 10% training data split.

VACA. Training for VACA was performed on normalized data—performing normalization only on the continuous variables, i.e. r.v. C and R in Fig 18. We trained a heterogeneous VACA as described in Section with a message passing function f^m with one hidden layer of 16 neurons, a decoder with one hidden layer of 16 neurons and a latent variable with dimension 4. We trained the model using the PIWAE (Rainforth et al. 2018) approach with $\lambda_{KLD} = 0.05$, specifically, the encoder with the IWAE (Burda, Grosse, and Salakhutdinov 2016) objective with $K = 5$ and the decoder with a β-ELBO with $\beta = 0.5$. We use a parents dropout rate (see Appendix E) of 0.2, learning rate of 0.005 and batch size 100.

Classifiers. Logistic Regresion (LR) and Support Vector Machine (SVM) classifiers are taken from the scikit-learn library and trained with default parameters as well as $\mathrm{class\_weight} = \mathrm{balanced}$ due to the class imbalance of the dataset.

Metrics

In this section we detail the measures f1-score (f1), unfairness (uf) and accuracy (acc) in Table 13.

f1-score. Due to class imbalance, we measure classifier performance with the f1-score. The f1-score is the weighted average of the precision and recall and can assume values between 0 and 1; the higher the values the better. Our implementation relies on the $\mathrm{f1\_score}$ from the scikit-learn library. We compute in expectation over our training dataset:

$$\mathrm{f1} = \mathbb{E}\left[2 \times \frac{\mathrm{precision} \times \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}\right]. \tag{16}$$

Metr.	CLF	full	unaware	fair-x	fair-z
↑ *acc*	LR	61.0	62.0	43.0	63.6 ± 4.74
	SVM	66.0	64.0	51.0	64.2 ± 4.52
↑ *f1*	LR	67.23	68.33	47.71	70.89 ± 4.11
	SVM	71.67	69.49	59.50	70.79 ± 5.15
↓ *uf*	LR	18.30 ± 3.22	17.65 ± 3.20	0.16 ± 0.02	0.44 ± 0.24
	SVM	14.01 ± 2.26	13.27 ± 2.28	0.14 ± 0.02	0.51 ± 0.19

Table 13: Counterfactual unfairness (*uf*), accuracy (*acc*) and f1-score (*f1*) of a LR and SVM classifier over 10 VACA seeds. Values multiplied by 100.

where $\mathrm{precision}$ is the ratio $\frac{TP}{TP+FP}$ with the number of true positives TP and the number of false positives FP and recall is the ratio $\frac{TP}{TP+FN}$ with the number of false positives FP.

Counterfactual (un)fairness. We measure counterfactual unfairness (Kusner et al. 2017) with counterfactual instances $\mathbf{x}^{CF}$ and classifier prediction $h(\mathbf{x}^{CF}) = \hat{y}^{CF}$ as expectation over our training dataset:

$$\begin{aligned}\mathrm{uf} = \mathbb{E}\big[\big|&p(y^F) = 1|\mathbf{x}^F) \\ &-p(\hat{y}^{CF} = 1|do(S = a'), \mathbf{x}^F)\big|\big]\end{aligned} \tag{17}$$

where $a' = 1 - a$ as $S \in \{0, 1\}$.

Accuracy. In Table 13, we report additionally the prediction accuracy as performance measure of classifier h with respect to factuals (samples) $(\mathbf{x}^F, y^F)$ and prediction $h(\mathbf{x}^F) = \hat{y}^F$ in expectation over our training dataset:

$$\mathrm{acc} = \mathbb{E}\left[\mathbb{I}\left({y^F}_i = \hat{y}_i^F\right)\right]. \tag{18}$$

Our implementation relies the $\mathrm{accuracy\_score}$ from the scikit-learn library.

Improving the interpretability of GNN predictions through conformal-based graph sparsification

PABLO SÁNCHEZ-MARTÍN, Max Planck Institute for Intelligent Systems, Germany and Saarland University, Germany

KINAAN AAMIR KHAN, McGill University, Canada

ISABEL VALERA, Saarland University, Germany and Max Planck Institute for Software Systems, Germany

Graph Neural Networks (GNNs) have achieved state-of-the-art performance in solving graph classification tasks. However, most GNN architectures aggregate information from all nodes and edges in a graph, regardless of their relevance to the task at hand, thus hindering the interpretability of their predictions. In contrast to prior work, in this paper we propose a GNN *training* approach that jointly i) finds the most predictive subgraph by removing edges and/or nodes—-*without making assumptions about the subgraph structure*—while ii) optimizing the performance of the graph classification task. To that end, we rely on reinforcement learning to solve the resulting bi-level optimization with a reward function based on conformal predictions to account for the current in-training uncertainty of the classifier. Our empirical results on nine different graph classification datasets show that our method competes in performance with baselines while relying on significantly sparser subgraphs, leading to more interpretable GNN-based predictions.

Additional Key Words and Phrases: Graph Neural Networks, Interpretability, Conformal Prediction, Reinforcement Learning

ACM Reference Format:

Pablo Sánchez-Martín, Kinaan Aamir Khan, and Isabel Valera. 2024. Improving the interpretability of GNN predictions through conformal-based graph sparsification. 1, 1 (March 2024), 26 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Graph Neural Networks (GNNs) have become a cornerstone in modern machine learning, excelling in diverse domains such as social network analysis [8, 9], recommender systems [3, 17] and bioinformatics [22, 38, 57]. However, the complexity of GNNs contributes to one of their principal shortcoming: a lack of human-interpretable predictions. This opacity hinders their potential for practical, real-world impact, as practitioners often require interpretable models to inform decision-making, ensure trustworthiness, and comply with regulations [16]. Current interpretability methods for GNN predictors aim to identify the most predictive subgraph from an original graph [46]. In essence, interpretability in this context is closely tied to graph sparsity, implying the use of a minimal set of nodes and edges from the graph for prediction, rather than the entire graph $\mathcal{G}$.

This viewpoint of interpretability is exemplified in Figure 1 with the synthetic BA2Shapes dataset [60]. In this dataset, a binary classification task can be solved using only specific motifs of the graphs (the house or the cycle motifs, i.e., yellow nodes), while the original graphs contain additional irrelevant information for the task at hand, i.e., purple nodes. A sparser graph reduces the volume of information used for making predictions, rendering it easier for humans to understand the

Authors' addresses: Pablo Sánchez-Martín, psanchez@tue.mpg.de, Max Planck Institute for Intelligent Systems, Tübingen, Germany and Saarland University, Saarbrücken, Germany; Kinaan Aamir Khan, McGill University, Montreal, Canada, kinaan.khan@mail.mcgill.ca; Isabel Valera, Saarland University, Saarbrücken, Germany and Max Planck Institute for Software Systems, Saarbrücken, Germany.

2024. Manuscript submitted to ACM

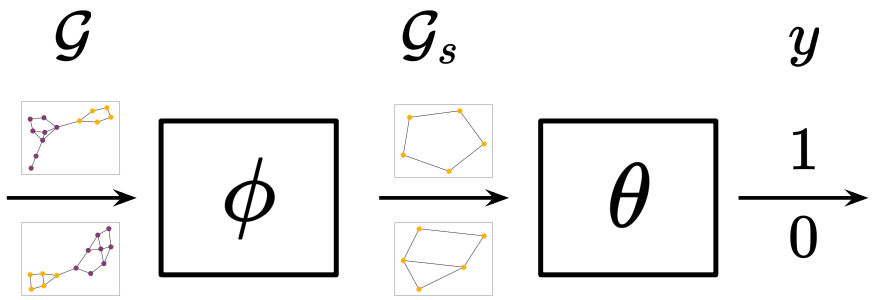


Fig. 1. **CORES pipeline.** Illustration of the pipeline of CORES using the synthetic BA2Shapes designed for binary graph classification. On the left, we present two examples of original graphs, denoted as $\mathcal{G}$, corresponding to the positive class (top, cycle motif) and negative class (bottom, house motif). The policy ϕ of CORES takes $\mathcal{G}$ and sparsifies it, resulting in the predictive subgraph $\mathcal{G}_s$, which retains only the relevant information for the task, i.e., the motifs. Finally, the graph classifier θ takes $\mathcal{G}_s$ as input and produces the prediction $y \in \{0, 1\}$.

GNN predictions. It is crucial to note that interpretability is achieved through sparsity only when the subgraph completely excludes information from the omitted nodes and edges since only then the subgraph, and thus, the explanation is *faithful* to the model prediction. This situation does not arise, for example, if message passing [25] is applied to obtain the node embeddings before finding the subgraph.

While most of explainability methods for GNNs are post-hoc, some existing approaches attempt to identify the predictive subgraph during training [7, 46]. These methods, however, exhibit two primary limitations: i) they often still rely on the entire graph for predictions [7], resulting in a lack of *faithfulness* to the model prediction (as described above), and/or ii) they impose strong assumptions concerning the structure of the predictive subgraph, such as a pre-defined size. In practical scenarios, relaxing assumptions on the subgraph structure is desirable. For instance, in community detection within social networks [11, 45], it is crucial to only remove edges connecting users across distinct communities but the number of inter-community connections varies significantly and is unknown a priori. Another relevant example is the prediction of mutagenic effects in compounds, where Debnath et al. [13] observed that diverse "motifs" of chemical elements are predictive. We refer the reader to Section 5 for a concrete example on the MUTAG dataset.

In this work, we introduce CORES, a novel algorithm for training GNNs that addresses the two fundamental challenges simultaneously: i) finding the informative subgraph $\mathcal{G}_s$ by removing edges and/or nodes, *without imposing assumptions on the subgraph's structure*, and ii) optimizing the performance of the graph classification task solely using the selected subgraph $\mathcal{G}_s$, thereby enhancing the interpretability of the classifier. To this end, we use a bi-level optimization approach. The optimization for performance adheres to the conventional supervised problem paradigm. We then apply reinforcement learning to identify the predictive subgraph, using a policy that allows edge and/or node removal modes. Our carefully designed reward function allows us to introduce inductive biases toward either sparse or high-performing solutions and uses conformal predictions [1] to account for classifier uncertainty at each training epoch.

In summary, our work presents the following key contributions: (i) A novel framework, CORES, which enables graph-level classification using only predictive subgraphs as input, leading to more interpretable solutions. (ii) The flexibility to choose between achieving sparsity through node or edge removal, which removes assumptions on the structure of the predictive subgraph and broadens the range of application scenarios. (iii) The design of a versatile reward function that takes into account the uncertainty of the classifier and allows to introduce inductive biases towards sparser

or more high-performing solutions. (iv) A comprehensive comparison across nine graph classification datasets, showing that CORES makes predictions using significantly sparser graphs while keeping competitive performance.

2 PRELIMINARIES

This section outlines necessary background information on the key building blocks of CORES: reinforcement learning, message-passing graph neural networks, and conformal predictions. We begin by introducing the notation used throughout this work.

Notation. A graph is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} \in \{1, \ldots, n\} = [n]$ is the set of n nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. The size of a set is represented as $|\mathcal{V}|$. A subgraph of $\mathcal{G}$ is denoted as $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s) \subseteq \mathcal{G}$ comprising subsets of nodes $\mathcal{V}_s \subseteq \mathcal{V}$ and edges $\mathcal{E}_s \subset \mathcal{E}$. A labeled dataset is represented as $\mathcal{D} = \{\mathcal{G}_i, \boldsymbol{y}_i\}_i$ where $\boldsymbol{y}_i$ denotes the target of $\mathcal{G}_i$. Here, without loss of generality, we focus on K classes graph classification, hence $\boldsymbol{y}_i \in \{1, \ldots, K\} = [K]$.

2.1 Reinforcement learning

Reinforcement Learning (RL) is a learning paradigm in which an agent learns to optimize decisions by interacting with an environment [48]. The objective is to find a policy π that maximizes the expected cumulative reward. An RL problem is typically modeled as a Markov Decision Process (MDP), defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathrm{R}, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, $\mathcal{P}$ the transition probability, R the reward function, and γ the discount factor.

Policy Gradient Methods. Policy gradient methods directly optimize the policy using gradient ascent on the expected cumulative reward [49]. Proximal Policy Optimization (PPO) [43] is a policy gradient method that introduces an objective function fostering exploration while mitigating drastic policy updates. PPO aims to solve the optimization problem:

$$L^{CLIP}(\phi) = \mathbb{E}_t \left[\min \left(r_t(\phi)\hat{A}_t, clip(r_t(\phi), 1-\epsilon, 1 \quad \epsilon)\hat{A}_t \right) \right] \tag{1}$$

where $r_t(\phi) = \frac{\pi_\phi(a_t|s_t)}{\pi_{\phi_{old}}(a_t|s_t)}$ is the probability ratio, $\hat{A}_t$ an estimator of the advantage function at time t, and ϵ a hyperparameter controlling the deviation from the old policy.

2.2 Graph neural networks

Message passing Graph Neural Networks (GNNs) are designed for graph-structured data processing. Each node $v \in \mathcal{V}$ has a feature vector $\boldsymbol{h}$, and the GNN transforms these features using neighborhood information. Formally, a GNN performs L update rounds, updating each node's feature vector $\boldsymbol{h}_i^{(l)}$ at step $l \geq 1$:

$$\boldsymbol{h}_i^{(l)} = \mathrm{f}_{\theta_u} \left(\bigoplus_{j \in \mathcal{N}_i} \mathrm{m}_{\theta_m}(\boldsymbol{h}_i^{(l-1)}, \boldsymbol{h}_j^{(l-1)}) \right). \tag{2}$$

Here, f_{θ_u} and m_{θ_m} are differentiable, parameterized functions, $\mathcal{N}_i$ represents node i's neighbors, and $\bigoplus$ denotes permutation invariant operations, i.e., sum, mean, or max. After L steps, we obtain the final node representations $\boldsymbol{h}_i^{(L)} \forall i \in \mathcal{V}$. These representations are used for tasks like graph-level prediction, employing a permutation-invariant readout function $\hat{\boldsymbol{y}} = \mathrm{g}\left(\left\{ \boldsymbol{h}_i^{(L)} : i \in \mathcal{V} \right\} \right)$ that ensures the output is node order independent.

2.3 Conformal Prediction

Conformal prediction [1] is a framework that rigorously quantifies uncertainty in machine learning predictions. Given a labeled dataset $\{x_i, y_i\}_{i=1}^N$, a heuristic measure of uncertainty of our predictor (e.g., softmax values from a classifier f_θ), and a scoring function $s(x, y) \in \mathbb{R}$ that reflects prediction uncertainty, conformal prediction generates a prediction set for a new input x_{test} as follows:

$$C\left(x_{\text{test}}\right) = \{y : s\left(x_{\text{test}}, y\right) \leq \hat{q}\} \subseteq [K]. \tag{3}$$

Here, $\hat{q}$ represents the $\frac{\lceil (n\ \ 1)(1-\alpha) \rceil}{n}$ quantile of the calibration scores $\{s_i\}_n$, where α is a predefined error rate. In the context of classification tasks, one commonly used conformal procedure is Adaptive Prediction Sets (APS) [2, 39], which defines the scoring function as:

$$s(x, y) = \sum_{j=1}^{k} \mathrm{f}_\theta(x)_{\pi_j(x)}, \text{ where } y = \pi_k(x) \tag{4}$$

Here, $\pi(x)$ represents the permutation of $[K]$ that arranges the softmax values $\mathrm{f}_\theta(x)$ from most likely to least likely.

3 RELATED WORK

GNNs extend deep learning models to incorporate graph-structured data [6]. Numerous architecture proposals have emerged, tailored for node, link, and graph prediction tasks [10, 12, 21, 41, 55, 56, 58]. However, these approaches rely on the complete graph data and often disregard interpretability, which is the focus of our work. More recently, some works aim for sparsification [23, 29, 30, 32, 37, 40, 53, 54, 64]. However, these methods typically make strong assumptions on the predictive subgraph. Moreover, they often focus on tasks other than graph prediction.

Here, we explore relevant sparsification techniques that specifically target supervised problems, which are the main focus of our paper. We categorize these works into two categories: those employing soft information removal, where the predictive subgraph retains information from the complete graph, and thus may not be faithful to the model predictions; and those adopting hard information removal, which closely aligns with our approach.

Soft removal of information. Several architectures [5, 24, 51] assign varying importance to network edges, simulating soft edge removal whilst maintaining information flow across nodes. DiffPool [59] hierarchically removes parts of the original graph, distilling the graph's representation into a single vector used for prediction. SAGPooling [26, 28], uses a GNN to update node features and subsequently selects the top k most relevant nodes based on their features. It is worth noting that even though part of the network is dropped, the remaining subgraph contains information from the entire network. In contrast, in our framework, the graph predictor does not have access to the dropped information.

Hard removal of information. GPool [7, 20, 26] selects the top k nodes based solely on their information. However, the number of nodes is fixed and cannot dynamically change for different graphs. SUGAR [46] takes a different approach by identifying discriminative subgraphs. It introduces a reinforcement pooling module trained with Q-learning [34] to adaptively select a pooling ratio when recombining subgraphs for classification. SUGAR also imposes a predefined size for the subgraphs. In contrast, CORES does not make any assumptions about the structure of the subgraph, uses policy gradient methods to capture uncertainty in the sparsity process, and allows for different modes of removal.

Finally, there is extensive literature that aims to understand which subgraph of the input graph is most relevant for making predictions in GNNs [31, 36, 42, 60–63]. However, it is important to note that these methods primarily focus on explaining a GNN predictor trained using the original graphs. In contrast, our objective is to train a GNN that inherently utilizes a minimal portion of the original graph. While these methods may complement our approach by providing further insights of the predictions, they operate in a post-hoc manner.

4 CORES: CONFORMAL-BASED REINFORCEMENT LEARNING FOR GRAPH SPARSIFICATION

In this section, we present CORES, a novel training procedure for GNNs leveraging reinforcement learning with conformal-based rewards to achieve graph sparsification. Given a labeled dataset $\mathcal{D}$, the main goal of CORES is to identify a compact predictive subgraph $\mathcal{G}_s \subseteq \mathcal{G}$ that maintains high performance for a graph classification task. Next, we provide a detailed description of the different parts comprising CORES.

4.1 Optimizing for sparsity and performance

Firstly, we aim to learn a function $\pi_\phi : \mathcal{G} \rightarrow \mathcal{G}_s$ responsible for identifying the predictive subgraph. This function corresponds to the policy of the reinforcement learning component of CORES. Secondly, we want to train a graph classifier $\mathrm{f}_\theta : \mathcal{G}_s \rightarrow \hat{\boldsymbol{y}}$ that takes the identified predictive subgraph as input and generates predictions. The pipeline of our approach is illustrated with the synthetic BA2Shapes dataset in Figure 1. To tackle this challenge we introduce a bi-level iterative optimization approach:

$$\phi^* = \arg\min_{\phi} \mathcal{L}_{\text{spa}} \left(\theta^\star(\phi), \phi, \mathcal{D}^{val} \right) \tag{5}$$

$$\text{s.t. } \theta^\star(\phi) = \arg\min_{\theta} \mathcal{L}_{\text{perf}} \left(\theta, \phi, \mathcal{D}^{tr} \right) \tag{6}$$

The nested structure of the problem implies that achieving an optimal predictive subgraph requires a high-performing graph classifier. We employ gradient descent for both optimizations, with a larger learning rate for the inner optimization [65]. Also, notice that we use the training $\mathcal{D}^{tr}$ and validation $\mathcal{D}^{val}$ sets to solve the inner and outer procedures respectively, which avoids overfitting. Algorithm 1 summarizes the training procedure.

Performance optimization. This objective corresponds to Equation (6) which represents a standard graph-supervised problem. Here, the objective is to minimize a loss function $\mathcal{L}_{\text{perf}}$, e.g., the cross-entropy loss. Thus the goal is to learn a function $\mathrm{f}_\theta : \mathcal{G}_s \rightarrow \hat{\boldsymbol{y}}$ with parameters θ that minimizes the prediction loss.

Sparsity optimization . This optimization task corresponds Equation (5). It presents a considerable challenge due to the combinatorial nature of the node and edge removal process. Specifically, we must determine both which edges ($(u, v) \in \mathcal{E}$) and/or nodes ($v \in \mathcal{V}$) and how many of them, should be removed. Drawing inspiration from SparRL [54], we formulate the sparsification task as a Markov Decision Process (MDP) and address it through the framework of graph reinforcement learning [35]: we parameterize the policy π_ϕ using a GNN. In contrast to prior approaches that rely on value-based methods like Deep Q-Learning [34], we opt for the policy gradient method Proximal Policy Optimization (PPO) [44] to capture the inherent uncertainty in the sparsification process. We denote the objective of sparsity optimization as $\mathcal{L}_{\text{spa}}$, which corresponds to solving the PPO objective in Equation (1). The outcome of this task is a policy π_ϕ that finds the

Algorithm 1 Training of CORES.

Input: Training $\mathcal{D}^{tr}$ and validation $\mathcal{D}^{val}$ sets; Initial graph classifier parameters θ and policy parameters ϕ; Empty buffer $\mathcal{B}$; Learning rates α and β; Number of PPO updates K
while not convergence **do**
  for $\{\boldsymbol{y}, \mathcal{G}\} \in \mathcal{D}^{tr}$ **do** ▷ Get a (batch of) sample(s) from the training set
    $\pi_\phi : \mathcal{G} \rightarrow \mathcal{G}_s$ ▷ Get the subgraph using the policy
    $\hat{\boldsymbol{y}}_s = \mathrm{f}_\theta(\mathcal{G}_s)$ ▷ Get the prediction
    $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{\text{perf}}(\theta, \phi)$ ▷ Update the parameters of the graph classifier
  end for
  for $\{\boldsymbol{y}, \mathcal{G}\} \in \mathcal{D}^{val}$ **do** ▷ Get a (batch of) sample(s) from the validation set
    $\pi_\phi : \mathcal{G} \rightarrow \mathcal{G}_s$ ▷ Use the π to sample a sparse graph
    $\hat{\boldsymbol{y}}_s = \mathrm{f}(\mathcal{G}_s)$ ▷ Get the prediction of the sparse graph
    Add tuple $\{\boldsymbol{y}, \mathcal{G}, \hat{\boldsymbol{y}}, \mathcal{G}_s\}$ to the buffer $\mathcal{B}$
  end for
  Compute the quantile $\hat{q}$ using the calibration scores obtained with $\mathcal{D}^{tr}$
  Compute rewards and prepare mini-batches from buffer $\mathcal{B}$ for PPO updates
  for i = $\{1, 2, \ldots, K\}$ **do** ▷ Update PPO K times
    $\{\boldsymbol{y}, \mathcal{G}, \hat{\boldsymbol{y}}, \mathcal{G}_s, r\} \sim \mathcal{B}$ ▷ Sample a batch of tuples from the buffer
    Compute advantage estimates $\hat{A}$
    $\phi \leftarrow \phi - \beta \nabla_\phi \mathcal{L}_{\text{spa}}(\theta, \phi)$ ▷ Update policy by maximizing Equation (1)
  end for
  Check for convergence, update convergence flag
end while
Output: Optimal policy parameters ϕ and graph classifier parameters θ

predictive subgraph $\mathcal{G}_s$, subsequently employed as input for the graph classification task. Next, we provide more details on the sparsity optimization.

4.2 Unpacking the sparsity optimization

In this subsection, we provide a detailed description of the components of the PPO method we designed to identify the predictive subgraph. In RL problems, agents usually make sequential decisions over multiple time steps, and the rewards they receive at each step may depend on the whole trajectory taken so far. This challenge, known as the "delayed reward" problem, is a central challenge in RL [47]. Our objective shifts towards identifying the predictive subgraph in a single step, rather than considering a trajectory as in traditional RL. This approach aims to find a strategy that maximizes the reward across the dataset graphs. This simplified scenario resembles the Multi-Armed Bandit Problem [27]. As our experiments in Section 5 empirically show, this simplified scenario effectively achieves high-performance relying on significantly sparser graphs. Below, we focus on describing two key components: the policy responsible for the removal process and the design of the reward function that guides the policy in finding the optimal predictive subgraph.

4.2.1 Policy formulation for graph sparsification via node or edge removal. We propose a policy $\pi(\boldsymbol{a}|\mathcal{G}; \phi)$ we model using a GNN with parameters ϕ. Given an input graph $\mathcal{G}$, the policy produces an action $\boldsymbol{a}$, where each element determines whether to keep ($a_i = 0$) or remove ($a_i = 1$) a specific node or edge in $\mathcal{V}$ or $\mathcal{E}$, respectively. This action results in a subgraph $\mathcal{G}_s \subseteq \mathcal{G}$. Recognizing the inherent uncertainty in real-world scenarios where the importance of a node/edge

may not be clear, we design the policy as a probability distribution. We establish two operational modes depending on the removal objective:

- **Node Removal Policy**: $\pi_n(\boldsymbol{a}|\mathcal{G};\phi)$, where $\boldsymbol{a} \in \{0,1\}^{|\mathcal{V}|}$. In this scenario, the resulting subgraph $\mathcal{G}_s$, used as input for the downstream task , satisfies $\mathcal{V}_s = \{v \in \mathcal{V} | a_v = 0\}$ and $\mathcal{E}_s = \{(u,v) \in \mathcal{E} | a_u = 0 \wedge a_v = 0\}$.
- **Edge Removal Policy**: $\pi_e(\boldsymbol{a}|\mathcal{G};\phi)$, where $\boldsymbol{a} \in \{0,1\}^{|\mathcal{E}|}$. Here, the subgraph $\mathcal{G}_s$, used as input for the downstream task, has $\mathcal{V}_s = \mathcal{V}$ and $\mathcal{E}_s = \{(u,v) \in \mathcal{E} | a_{uv} = 0\}$.

Note that node removal implies the removal of all edges connected to the node. In contrast, the edge removal policy retains all nodes, making it suitable for node classification, which is a direction for future work. In either case, determining $\mathcal{G}_s$ poses an NP-hard problem for both removal modes. The number of potential subgraphs increases exponentially with the number of nodes/edges in the original graph. Specifically, the number of possible actions for π_n is $\sum_{i=1}^{|\mathcal{V}|-1} \binom{|\mathcal{V}|}{i}$, while for π_e, it is $\sum_{i=1}^{|\mathcal{E}|} \binom{|\mathcal{E}|}{i}$. RL can be helpful to solve combinatorial problems, particularly when the reward function effectively guides the optimization process [33].

4.2.2 Reward formulation. The reward function R plays a pivotal role in shaping the policy's behavior. It should provide positive rewards for predictive and sparse subgraphs while penalizing those that negatively impact the performance of the graph classifier. It is also important to consider that the classifier makes errors, i.e., it does not achieve perfect performance. Consequently, there are instances where a subgraph may be genuinely predictive, but the classifier produces an incorrect prediction. Our reward design accounts for this inherent uncertainty through the use of conformal predictions, as described in a subsequent section. It consists of two primary components: one aimed at enhancing performance and the other dedicated to promoting sparsity.

Performance Reward (R_p). Our approach to rewarding performance is straightforward. We simply use the softmax score assigned by the graph classifier to the correct class y, i.e., $\mathrm{R}_p = \mathrm{f}_\theta(\mathcal{G}_s)_y \in [0,1]$.

Sparsity Reward (R_s). To incentivize a minimal predictive subgraph, we introduce a component that penalizes the *nodes ratio* $\mathrm{PR}_n = \frac{|\mathcal{V}_s|}{|\mathcal{V}|}$ or the *edges ratio* $\mathrm{PR}_e = \frac{|\mathcal{E}_s|}{|\mathcal{E}|}$ kept from the original graph, depending on the policy mode. To control the desired sparsity level, we introduce a parameter $d \in [0,1]$ representing the *maximum desired nodes/edges ratio*. Then, we define the sparsity reward as $\mathrm{R}_s(\mathcal{G}_s) = 1 - \mathrm{PR}^{\tilde{d}}$, where $\tilde{d}$ is a transformation of d such that $\mathrm{R}_s = 0.95$ when $\mathrm{PR} = d$. The variation of R_s with respect to PR for different d values is depicted in the inline figure. Each vertical line represents a unique d value, extending upwards until $\mathrm{R}_s = 0.95$ for all instances.

To seek sparsity, a smaller d value (e.g., blue or yellow curves) is preferred. This leads to a slower increase in R_s with node/edge removal, rewarding substantial removal. Conversely, when d approaches 1 (e.g., pink curve), R_s close to 1 are awarded even for keeping most nodes/edges.

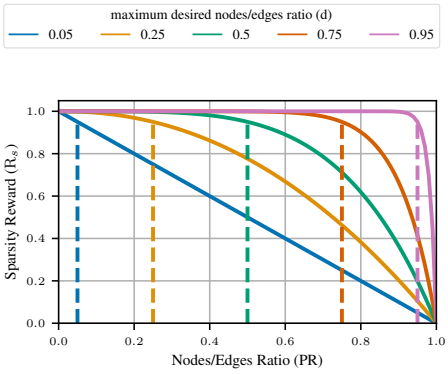


Fig. 2. Evolution of the sparsity reward R_s over the node/edge ratio for different values of the *maximum desired nodes/edges ratio* d.

Note that both reward components fall within the range $[0, 1]$, facilitating comparison. Now, we proceed to define the reward function used in CORES:

$$\mathrm{R} = \begin{cases} \lambda \mathrm{R}_p \quad (1-\lambda)\mathrm{R}_s(\mathcal{G}_s) & \text{if } y \in C(\mathcal{G}_s) \wedge |C(\mathcal{G}_s)| = 1, \\ \frac{\mathrm{R}_p}{|C(\mathcal{G}_s)|} & \text{if } y \in C(\mathcal{G}_s) \wedge |C(\mathcal{G}_s)| > 1, \\ -\mathrm{R}_s(\mathcal{G}_s) & \text{if } y \quad C(\mathcal{G}_s). \end{cases} \tag{7}$$

There are two key aspects to highlight. Firstly, we compute the quantile $\hat{q}$ used to obtain the prediction set C using the scoring function introduced in Equation (4). Secondly, it involves three different cases based on whether the true label y is within the prediction set and the size of the prediction set.

The top scenario corresponds to cases in which the predictive subgraph leads to a highly certain and correct prediction. Here, a parameter $\lambda \in [0, 1]$ allows practitioners to emphasize either performance ($\lambda = 1$) or sparsity ($\lambda = 0$). In this scenario, the reward is in the range $\mathrm{R} \in [0, 1]$. The middle scenario addresses instances in which the subgraph leads to the classifier being uncertain about the prediction, i.e., $|C(\mathcal{G}_s)| > 1$. In such cases, we aim to provide a positive but smaller reward within the range $(0, 0.5)$. This reward encourages to increase the probability of the true label and to reduce the size of the prediction set, that is moving to the top scenario. Intuitively, this scenario arises with challenging examples. The bottom scenario refers to cases in which the predictive subgraph results in an incorrect prediction, falling outside the prediction set C. As this behavior is undesirable, the rewards R fall within the range $[-1, 0]$, encouraging to reduce sparsity. In this scenario, the reward reaches 0 only when we recover the original graph.

The use of conformal prediction in the reward function is essential to guide the policy toward discovering a good predictive subgraph. If the task is challenging and the classifier exhibits uncertainty, the focus should initially be on performance. Only when the classifier is confident about the prediction should we encourage sparsity. In the following section, we present empirical evidence supporting the reward function's design, and more generally our approach CORES, for achieving both performance and compact predictive subgraphs.

5 EXPERIMENTS

In this section, we present a comprehensive set of experiments to assess the performance of CORES. First, we conduct an ablation study to show the impact of different values of λ and maximum desired ratio (d) on performance, as well as the effect of the choice of base GNN architecture. Then, we compare CORES with relevant baselines.

Proposed approaches. We evaluate the two information removal modes of the proposed framework, described in Section 4.2.1. We denote the model with the policy that removes nodes as $\mathrm{CORES}_{\mathrm{N}}$ and the model with the policy that exclusively removes edges as $\mathrm{CORES}_{\mathrm{E}}$.

Datasets. In this study, we used seven Bioinformatics datasets, which include MUTAG [14], DD [15], ENZYMES [4], NCI1 [52], NCI109 [52], PTC [50], and PROTEINS [4]. Additionally, we incorporated two chemical compound datasets, BZR [18] and COX2 [18]. Table 1 shows the statistics of the datasets.

Experimental setup. Aiming for computational efficiency, we conducted cross-validation on the hyperparameters of the vanilla GNN classifier, i.e., the GNN architecture (e.g., GIN) trained with stochastic gradient descent. We selected the optimal configuration of hyperparameters based on the validation set and used it to train the remaining models. We

Table 1. **Statistics of the datasets.**

Dataset	# Graphs	# Features	# Edge Features	# Classes	Undirected	# Nodes	# Edges
BZR	405	10	0	2	Yes	35.75 ± 7.26	76.72 ± 15.39
COX2	467	10	0	2	Yes	41.22 ± 4.03	86.89 ± 8.53
DD	1178	20	0	2	Yes	284.32 ± 272.00	1431.32 ± 1387.81
ENZYMES	600	21	0	6	Yes	32.63 ± 15.28	124.27 ± 51.00
MUTAG	135	7	4	2	Yes	18.85 ± 4.49	41.67 ± 11.13
NCI1	4110	7	0	2	Yes	29.87 ± 13.56	64.60 ± 29.87
NCI109	4127	8	0	2	Yes	29.68 ± 13.57	64.26 ± 29.92
PROTEINS	1113	4	0	2	Yes	39.06 ± 45.76	145.63 ± 169.20
PTC	349	8	4	2	Yes	14.11 ± 8.44	28.97 ± 18.88

performed 5-fold cross-validation and reported standard deviation values on the test set at the last epoch. All experiments were conducted on a single CPU with 8GB of RAM. Here, we present results obtained using the GIN architecture. Additional results using other well-known architectures, namely GAT and GCN, are provided in Appendix C. For a comprehensive overview of the best configuration hyperparameters obtained for each model, architecture and dataset, please refer to Appendix A. We implemented the baselines using the Torch Geometric package [19] whenever it was available. Furthermore, we provide the implementation of CORES, along with the necessary scripts to replicate the experiments, at https://github.com/XXXX/XXXX.

5.1 Ablation study

In this section, we provide an ablation study on the proposed framework analyzing the effects of key hyperparameters and different choices for the base GNN architecture.

Analysis of the impact of λ and d. The reward function, presented in Equation (7), leverages two hyperparameters: the maximum desired nodes/edges ratio (d) and the balancing parameter (λ) to prioritize performance or sparsity. For the nine datasets under study, we present the effect of varying d and λ in Figure 3 and Figure 4, respectively, for both CORES_{N} (solid lines) and CORES_{E} (dashed lines). For this ablation study, we use GIN as the base GNN architecture. The results are color-coded for each dataset. The figures in the top row contain the performance analysis, while the middle row shows the sparsity, specifically the node ratio ($\frac{|\mathcal{V}_s|}{|\mathcal{V}|}$) for CORES_{N} and the edge ratio ($\frac{|\mathcal{E}_s|}{|\mathcal{E}|}$) for CORES_{E}. The bottom row shows performance versus sparsity in the vertical and horizontal axes respectively.

In Figure 3, we fix $\lambda = 0.0$ (only rewarding sparsity) while varying d within the range $[0.2, 0.9]$. Regarding performance (top row), we observe that generally, the value of d does not have an impact. The middle row, which focuses on sparsity, reveals interesting variations among datasets. For instance, for the MUTAG dataset, CORES finds a predictive subgraph that utilizes as little as 20% of the original nodes/edges with $d = 0.2$. In contrast, for the PTC dataset, we keep approximately 80% of the original nodes in CORES_{N} and 60% of the original edges in CORES_{E}, even with low d values. The bottom figures highlight that for certain datasets, e.g., BZR, CORES can maintain comparable accuracy regardless of sparsity levels. Conversely, for other datasets, decreasing sparsity tends to enhance performance, such as the case with the PTC dataset and CORES_{N}. These findings suggest that some datasets exhibit rapid performance degradation when information is removed. Examining the different reward scenarios in our reward definition (refer to Equation (7)), it is

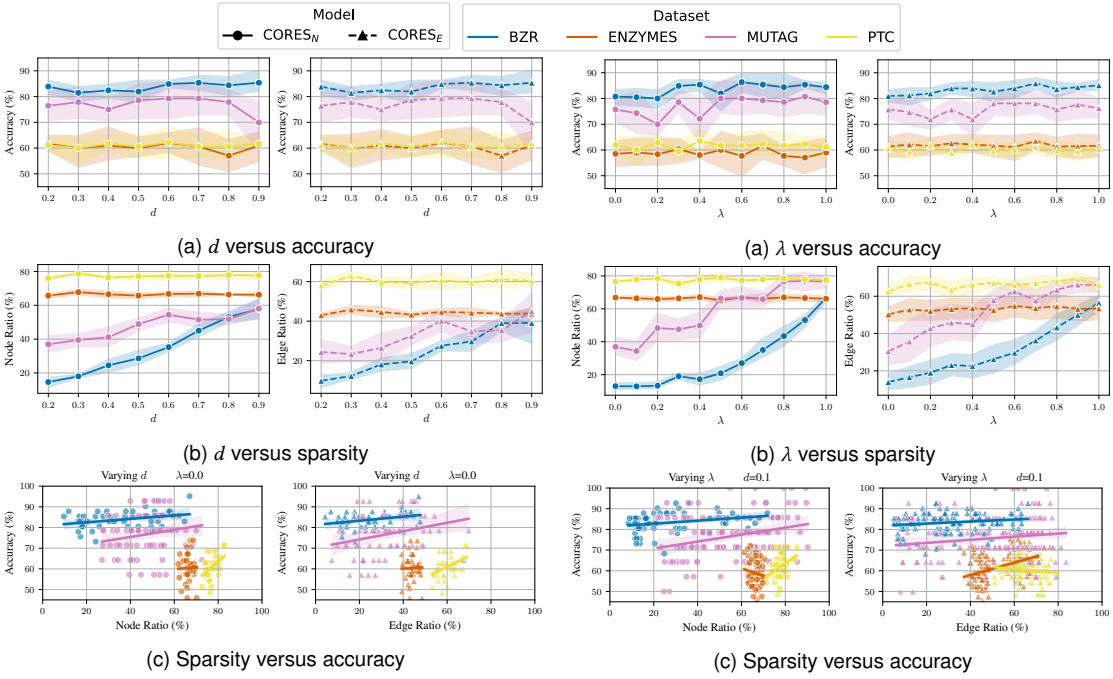


Fig. 3. **Ablation study on the maximum desired ratio d.** We use the GIN architecture and run each experiment for 5 different folds.

Fig. 4. **Ablation study on λ.** We use the GIN architecture and run each experiment for 5 different folds.

intuitive to deduce that CORES will prioritize reducing sparsity if the prediction is not within the prediction set (bottom scenario) or enhancing performance when the classifier exhibits uncertainty (middle scenario).

In Figure 4, we fix $d = 0.1$ (favoring high levels of sparsity) and we vary λ within the range $[0.0, 1.0]$. We observe a similar behavior as in Figure 3. The top row consistently shows that varying λ has a minimal impact on accuracy, and only sometimes leads to increased accuracy as λ increases (e.g., BZR dataset). Importantly, for none of the datasets does performance degrade with higher λ values. This behavior aligns with our objective, as higher values of λ correspond to a greater emphasis on rewarding performance. The middle row reveals two distinct patterns: i) a high correlation between λ and the number of nodes/edges kept, particularly evident in datasets like MUTAG, and ii) minimal sensitivity, as observed with the PTC dataset. The bottom row reveals similar patterns: positive correlation between the node/edge ratio and accuracy. It is also worth noting that varying λ results in a wide range of sparsity levels for some datasets, such as 20-90% for MUTAG, and a narrower range for others, like 40-70% for PTC.

In summary, we observe that varying d or λ leads to one of two scenarios, depending on the dataset: i) A minimal change in performance and low sparsity, which suggests that information removal significantly degrades performance for these datasets. ii) A small positive correlation between the hyperparameters and performance and a high negative correlation between the hyperparameters and sparsity. Based on these insights, we recommend that practitioners use d to fine-tune sparsity levels and consider $\lambda = 1.0$ when prioritizing performance over sparsity. Appendix B contains the results for the rest of the datasets under study.

Table 2. **Ablation study on GNN architecture.** We compare the performance (accuracy) and sparsity (node/edge ratio) of CORES with the Vanilla approach for the GCN (left) and GAT(right) architectures. We show the mean over 5 independent runs and the standard deviation as the subindex. All metrics are shown in percentages.

Dataset	Metric	GCN			GAT		
		Vanilla	**CORES$_N$**	**CORES$_E$**	**Vanilla**	**CORES$_N$**	**CORES$_E$**
BZR	**Accuracy**	$86.83_{4.08}$	$82.44_{4.69}$	$80.98_{5.29}$	$80.98_{4.36}$	$83.90_{4.43}$	$81.46_{4.08}$
	Node Ratio	-	$76.98_{6.06}$	$100.00_{0.00}$	-	$37.99_{10.27}$	$100.00_{0.00}$
	Edge Ratio	-	$61.17_{11.31}$	$46.16_{5.68}$	-	$29.92_{8.83}$	$57.26_{21.50}$
COX2	**Accuracy**	$79.15_{4.85}$	$80.85_{3.01}$	$81.70_{4.90}$	$82.13_{4.15}$	$81.70_{4.41}$	$80.85_{2.61}$
	Node Ratio	-	$72.49_{4.24}$	$100.00_{0.00}$	-	$77.91_{2.29}$	$100.00_{0.00}$
	Edge Ratio	-	$53.19_{8.07}$	$31.27_{23.60}$	-	$62.62_{3.24}$	$66.18_{0.38}$
DD	**Accuracy**	$78.47_{3.47}$	$79.32_{2.58}$	$75.76_{4.05}$	$76.10_{3.02}$	$75.42_{4.19}$	$76.44_{2.19}$
	Node Ratio	-	$56.44_{2.50}$	$100.00_{0.00}$	-	$75.16_{1.39}$	$100.00_{0.00}$
	Edge Ratio	-	$32.01_{2.89}$	$51.39_{1.90}$	-	$56.42_{2.16}$	$58.62_{2.51}$
ENZYMES	**Accuracy**	$66.00_{5.35}$	$58.31_{8.47}$	$57.02_{11.62}$	$71.33_{10.63}$	$69.84_{4.43}$	$67.54_{12.23}$
	Node Ratio	-	$77.17_{7.28}$	$100.00_{0.00}$	-	$80.26_{1.03}$	$100.00_{0.00}$
	Edge Ratio	-	$62.33_{9.26}$	$74.20_{35.99}$	-	$65.92_{1.80}$	$61.60_{9.06}$
MUTAG	**Accuracy**	$81.43_{3.91}$	$81.43_{10.83}$	$87.14_{6.56}$	$81.43_{6.39}$	$80.00_{5.98}$	$81.43_{6.02}$
	Node Ratio	-	$79.94_{2.79}$	$100.00_{0.00}$	-	$75.72_{4.04}$	$100.00_{0.00}$
	Edge Ratio	-	$63.52_{4.23}$	$32.66_{8.79}$	-	$58.74_{6.05}$	$55.25_{14.77}$
NCI1	**Accuracy**	$77.33_{2.09}$	$66.45_{7.71}$	$65.94_{6.31}$	$79.42_{1.34}$	$73.43_{2.67}$	$75.38_{1.22}$
	Node Ratio	-	$76.47_{17.88}$	$100.00_{0.00}$	-	$63.49_{3.58}$	$100.00_{0.00}$
	Edge Ratio	-	$64.02_{18.84}$	$37.98_{14.34}$	-	$53.33_{2.81}$	$88.68_{6.67}$
NCI109	**Accuracy**	$79.32_{1.57}$	$76.08_{2.71}$	$77.14_{2.02}$	$76.90_{1.34}$	$74.72_{2.25}$	$72.78_{2.29}$
	Node Ratio	-	$78.78_{4.15}$	$100.00_{0.00}$	-	$71.55_{11.30}$	$100.00_{0.00}$
	Edge Ratio	-	$67.08_{4.42}$	$92.02_{1.93}$	-	$62.37_{12.96}$	$62.37_{13.13}$
PROTEINS	**Accuracy**	$72.86_{1.02}$	$73.21_{2.45}$	$75.89_{1.94}$	$75.36_{3.00}$	$73.39_{3.86}$	$73.57_{1.62}$
	Node Ratio	-	$74.66_{6.10}$	$100.00_{0.00}$	-	$67.94_{2.97}$	$100.00_{0.00}$
	Edge Ratio	-	$56.12_{9.77}$	$64.73_{0.98}$	-	$46.56_{4.27}$	$52.82_{1.86}$
PTC	**Accuracy**	$57.78_{7.71}$	$62.86_{2.86}$	$63.43_{3.13}$	$64.00_{4.33}$	$62.96_{4.17}$	$59.43_{4.69}$
	Node Ratio	-	$64.10_{4.19}$	$100.00_{0.00}$	-	$45.61_{6.00}$	$100.00_{0.00}$
	Edge Ratio	-	$40.21_{6.93}$	$44.47_{5.29}$	-	$21.24_{6.53}$	$76.36_{14.75}$

Ablation on the base GNN architecture. Now, we analyze how switching the base architecture affects the performance of CORES. Table 2 summarizes the results for two base architectures: GCN and GAT. We observe that even when the base architecture changes, both CORES$_N$ and CORES$_E$ achieve competitive accuracy compared to the respective Vanilla approach that simply trains the GNN architecture using all the information from the original graph. We refer the reader to Appendix C for the complete results.

5.2 Performance comparison

In this section, we conduct a comprehensive evaluation of CORES by comparing it with baselines and competing methods across nine datasets.

We assess the performance on the test set using three key metrics: accuracy (the larger the better ↑), as well as the percentage of nodes (for CORES_N) or edges (for CORES_E) kept in the subgraph (the lower the better ↓). Decreasing these percentages increases graph sparsity, enhancing model interpretability.

Baselines. We divided the baselines into two categories: *Full* model (use complete graph information) and *Sparse* (use part of the graph information) model approaches. In the *Full* model category, we compare with the following approaches: Vanilla approach representing training the base GNN architecture without any sparsity consideration, DiffPool [59], and SAGPooling [26, 28]—which uses node embeddings coming from a GNN to select the top k nodes, thus incorporating global information. For the *Sparse* models, we compare with: SUGAR [46] [1] (for the datasets that their code can run) and GPool [7, 20, 26]— where the node embeddings used to select the top k come from a multi-layer perceptron, hence the subgraph does not contain global information. Here, we present the results using GIN [58] as the base GNN architecture. Refer to Appendix C for the results using GCN and GAT.

Quantitative results. Table 3 summarizes the results. In terms of accuracy, we observe that while Vanilla consistently achieves the highest accuracy on most datasets, CORES_E and CORES_N secure the second and third-best positions, outperforming all other *Full* models and all *Sparse* models, which highlights the competitive performance of the proposed approach. On the other hand, SUGAR consistently underperforms across all datasets, and its results exhibit high variance. Regarding interpretability (understood as sparsity), we perform a node and edge ratio analysis on the three best-performing *Sparse* models in terms of accuracy: CORES_N, CORES_E, and GPool. It is important to note that CORES_E consistently maintains a node ratio of 100%, due to its edge removal strategy that preserves all nodes. Among these models, CORES_N achieves the highest rankings in both node and edge ratios, indicating its effectiveness in removing a larger number of nodes from the original graph. Consequently, the graph classifier operates on a reduced, and thus more interpretable, subgraph. Finally, we find it interesting to analyze the results for the PTC dataset (bottom row). We observe that CORES, in both removal modes, achieves the worst sparsity results but at the same time the best performance among all models. In terms of sparsity, CORES_N keeps 96.02% of the nodes, while CORES_E keeps 88.61% of the edges. In terms of accuracy, all the models achieve less than 65%, which is notably low for a binary classification problem. This indicates that the classifier makes a substantial number of errors. This observation is important considering the design of the proposed reward function of CORES (see Equation (7)). For this dataset, the reward function places CORES in a scenario where it seeks to enhance performance, corresponding to the middle case, or penalize sparsity, corresponding to the bottom case of the reward function. In essence, CORES behaves in accordance with the reward function.

Qualitative results. Figure 5 presents two examples of the predictive subgraphs discovered by CORES_N for the MUTAG dataset, showcasing positive (top) and negative (bottom) instances. In this dataset, the positive class ($y = 1$) represents mutagenic molecules, while the negative class ($y = 0$) corresponds to non-mutagenic molecules. The left side of the figure shows the original graphs, while the center illustrates the predictive subgraphs identified by CORES_N. We observe that for the mutagenic graph (top), CORES_N keeps $N0_2$ nitrogen dioxide compounds and some carbon atoms, while for the non-mutagenic graph (bottom), it captures chlorine carbon atoms as relevant for the classification task. These findings align with prior in the chemical domain [13].

[1] We use the official implementation of SUGAR to extract the results on the above datasets. We believe our evaluation of a held-out test set may explain the disparity between the results we report and those in [46]. Instead, the available implementation of SUGAR evaluates on the same validation set used to select the best model.

Table 3. **Model comparison results.** We show the mean over 5 independent runs and the standard deviation as the subindex. All metrics are shown in percentage. The last rows include the average ranking of the model across datasets. Best performing models on average are indicated in bold.

Dataset	Metric	*Full* Models			*Sparse* Models			
		Vanilla	**SAGPool**	**DiffPool**	**SUGAR**	**GPool**	**CORES$_N$**	**CORES$_E$**
BZR	**Accuracy**	$82.44_{5.00}$	$85.85_{2.67}$	$70.24_{5.95}$	-	$80.98_{4.01}$	$84.88_{4.01}$	$81.95_{5.62}$
	Node Ratio	-	-	-	-	$96.46_{0.19}$	$15.91_{5.30}$	$100.00_{0.00}$
	Edge Ratio	-	-	-	-	$96.40_{0.45}$	$10.84_{5.80}$	$76.03_{3.72}$
COX2	**Accuracy**	$82.13_{5.12}$	$80.43_{3.16}$	$65.83_{6.65}$	-	$80.43_{3.50}$	$82.98_{3.36}$	$85.11_{3.01}$
	Node Ratio	-	-	-	-	$91.15_{0.14}$	$86.44_{3.48}$	$100.00_{0.00}$
	Edge Ratio	-	-	-	-	$89.20_{1.08}$	$77.63_{6.60}$	$58.30_{28.00}$
DD	**Accuracy**	$78.81_{3.44}$	$74.07_{4.09}$	-	-	$77.46_{3.31}$	$75.42_{5.22}$	$75.25_{2.77}$
	Node Ratio	-	-	-	-	$40.23_{0.03}$	$47.53_{15.04}$	$100.00_{0.00}$
	Edge Ratio	-	-	-	-	$15.41_{0.26}$	$25.88_{14.79}$	$51.55_{2.42}$
ENZYMES	**Accuracy**	$61.67_{8.50}$	$45.90_{5.18}$	$31.00_{6.66}$	$16.67_{23.57}$	$62.62_{5.36}$	$62.33_{6.08}$	$64.92_{3.40}$
	Node Ratio	-	-	-	-	$96.67_{0.14}$	$80.67_{1.16}$	$100.00_{0.00}$
	Edge Ratio	-	-	-	-	$93.37_{0.14}$	$66.90_{1.66}$	$72.71_{1.00}$
MUTAG	**Accuracy**	$85.71_{8.75}$	$78.57_{5.05}$	$69.00_{12.21}$	$76.34_{3.80}$	$81.43_{3.91}$	$82.86_{6.39}$	$82.14_{5.98}$
	Node Ratio	-	-	-	-	$51.38_{0.12}$	$52.38_{7.67}$	$100.00_{0.00}$
	Edge Ratio	-	-	-	-	$36.56_{9.20}$	$36.31_{11.32}$	$61.09_{12.39}$
NCI1	**Accuracy**	$76.89_{2.03}$	$69.29_{2.12}$	$69.93_{3.24}$	$49.95_{35.58}$	$77.77_{1.93}$	$73.34_{6.30}$	$74.34_{7.53}$
	Node Ratio	-	-	-	-	$96.89_{0.08}$	$89.13_{13.88}$	$100.00_{0.00}$
	Edge Ratio	-	-	-	-	$94.47_{0.28}$	$86.61_{20.20}$	$84.06_{27.86}$
NCI109	**Accuracy**	$78.64_{2.69}$	$74.14_{0.85}$	$67.49_{2.60}$	$49.65_{35.71}$	$75.45_{3.30}$	$73.03_{1.90}$	$75.74_{2.04}$
	Node Ratio	-	-	-	-	$96.79_{0.04}$	$66.31_{5.69}$	$100.00_{0.00}$
	Edge Ratio	-	-	-	-	$93.65_{0.22}$	$61.26_{7.22}$	$71.34_{7.62}$
PROTEINS	**Accuracy**	$75.18_{2.04}$	$70.54_{3.63}$	$66.42_{7.19}$	$59.57_{43.01}$	$72.68_{1.85}$	$73.75_{4.02}$	$74.11_{2.82}$
	Node Ratio	-	-	-	-	$41.92_{0.30}$	$87.01_{20.04}$	$100.00_{0.00}$
	Edge Ratio	-	-	-	-	$19.42_{0.64}$	$78.55_{32.05}$	$83.31_{37.32}$
PTC	**Accuracy**	$63.43_{1.28}$	$61.14_{8.23}$	$54.44_{11.10}$	$56.14_{8.95}$	$59.43_{3.73}$	$64.00_{3.26}$	$63.43_{1.28}$
	Node Ratio	-	-	-	-	$43.52_{0.67}$	$96.02_{8.80}$	$100.00_{0.00}$
	Edge Ratio	-	-	-	-	$24.80_{3.48}$	$93.19_{15.13}$	$88.61_{15.64}$
Avg. Rank	**Accuracy**	2.11	4.44	6.12	6.67	3.44	2.67	2.67
	Node Ratio	-	-	-	-	1.56	1.44	-
	Edge Ratio	-	-	-	-	2.22	1.67	2.11

Time complexity. Training and inference are significantly faster in methods that employ a straightforward forward pass compared to RL-based methods like SUGAR and CORES. Notably, our findings indicate that CORES is at least as fast as SUGAR during training and can be up to 10 times faster during inference. For comprehensive quantitative results across all datasets, please refer to Appendix D.

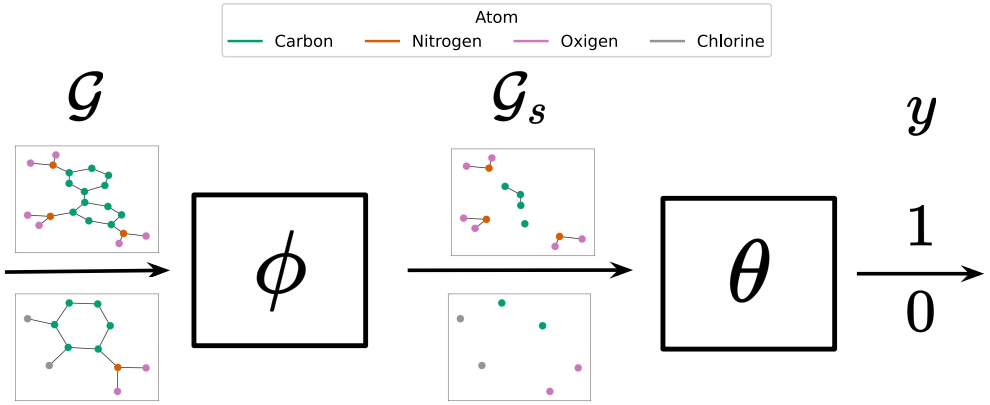


Fig. 5. Illustration of the predictive subgraph obtained by CORES_{N} for a mutagenic (top) and non-mutagenic (bottom) graph of the MUTAG dataset.

6 CONCLUSIONS

Graph Neural Networks (GNNs) have demonstrated exceptional performance in graph-level tasks, but suffer from interpretability issues due to their complexity. Existing interpretability research in GNNs mainly focuses on post-hoc explanations. Approaches that aim to introduce *sparsity* during training often provide predictive subgraphs that are not *faithful* with the classifier predictions. Also, these methods often rely on complete graph information and/or make strong assumptions about the sparser graphs. In this work, we introduced CORES, a novel GNN training approach that simultaneously identifies predictive subgraphs by removing edges and/or nodes, without imposing assumptions about subgraph structures; and optimizes the performance of the graph classification task. The optimization for performance refers to the conventional supervised problem paradigm. We then leverage reinforcement learning to identify predictive subgraphs, using a policy that allows both edge and node removal modes. One key aspect of CORES is the design of the reward function. This function allows the practitioner to introduce inductive biases towards either sparse or high-performing predictive subgraphs and it incorporates conformal predictions [1] to account for classifier uncertainty. Our empirical evaluation, conducted on nine graph classification datasets, provides evidence that our approach not only competes in performance with the baselines that use the complete graph information but also relies on significantly sparser subgraphs. Consequently, the resulting GNN-based predictions are more interpretable, addressing the primary motivation behind our work.

Practical limitations. The main limitation of our proposed approach lies in the significantly increased training and inference time, introduced by the reinforcement learning component, compared to the baselines that do not use reinforcement learning.

Future work. Future research could focus on improving the efficiency of the reinforcement learning component to accelerate the training process of CORES. Another interesting avenue for future work could involve modifying the current reward function to penalize specific types of errors or undesirable structures, such as isolated nodes. Additionally, it could also be interesting to explore the application of CORES to graph regression tasks and CORES_{E} to node classification tasks.

REFERENCES

[1] Anastasios Nikolas Angelopoulos and Stephen Bates. 2021. A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification. *ArXiv* abs/2107.07511 (2021). https://api.semanticscholar.org/CorpusID:235899036

[2] Anastasios Nikolas Angelopoulos, Stephen Bates, Jitendra Malik, and Michael I. Jordan. 2020. Uncertainty Sets for Image Classifiers using Conformal Prediction. *ArXiv* abs/2009.14193 (2020). https://api.semanticscholar.org/CorpusID:221995507

[3] Ting Bai, Youjie Zhang, Bin Wu, and Jian-Yun Nie. 2020. Temporal Graph Neural Networks for Social Recommendation. *2020 IEEE International Conference on Big Data (Big Data)* (2020), 898–903.

[4] Karsten M. Borgwardt, Cheng Soon Ong, Stefan Schönauer, S. V. N. Vishwanathan, Alex Smola, and Hans-Peter Kriegel. 2005. Protein function prediction via graph kernels. *Bioinformatics* 21 Suppl 1 (2005), i47–56.

[5] Shaked Brody, Uri Alon, and Eran Yahav. 2021. How Attentive are Graph Attention Networks? *ArXiv* abs/2105.14491 (2021).

[6] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur D. Szlam, and Pierre Vandergheynst. 2016. Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine* 34 (2016), 18–42.

[7] Cătălina Cangea, Petar Velickovic, Nikola Jovanovic, Thomas Kipf, and Pietro Lio'. 2018. Towards Sparse Hierarchical Graph Classifiers. *ArXiv* abs/1811.01287 (2018). https://api.semanticscholar.org/CorpusID:53219108

[8] Qi Cao, Huawei Shen, Jinhua Gao, Bingzheng Wei, and Xueqi Cheng. 2019. Coupled Graph Neural Networks for Predicting the Popularity of Online Content. *ArXiv* abs/1906.09032 (2019).

[9] Qi Cao, Huawei Shen, Jinhua Gao, Bingzheng Wei, and Xueqi Cheng. 2019. Popularity Prediction on Social Platforms with Coupled Graph Neural Networks. *Proceedings of the 13th International Conference on Web Search and Data Mining* (2019).

[10] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and Deep Graph Convolutional Networks. In *International Conference on Machine Learning*.

[11] Martina Contisciani, Eleanor A. Power, and Caterina De Bacco. 2020. Community detection with node attributes in multilayer networks. *Scientific Reports* 10 (2020).

[12] Gabriele Corso, Luca Cavalleri, D. Beaini, Pietro Lio', and Petar Velickovic. 2020. Principal Neighbourhood Aggregation for Graph Nets. *ArXiv* abs/2004.05718 (2020).

[13] Asim Kumar Debnath, Rosa L. Lopez de Compadre, Gargi Debnath, Alan J. Shusterman, and Corwin Hansch. 1991. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. Correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry* 34 2 (1991). https://api.semanticscholar.org/CorpusID:19990980

[14] Asim Kumar Debnath, R L Lopez de Compadre, Gargi Debnath, Alan J. Shusterman, and Corwin Hansch. 1991. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. Correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry* 34 2 (1991), 786–97.

[15] Paul D. Dobson and Andrew J. Doig. 2003. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology* 330 4 (2003), 771–83.

[16] Finale Doshi-Velez and Been Kim. 2017. Towards A Rigorous Science of Interpretable Machine Learning. *arXiv: Machine Learning* (2017).

[17] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Yihong Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph Neural Networks for Social Recommendation. *The World Wide Web Conference* (2019).

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

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

[20] Hongyang Gao and Shuiwang Ji. 2019. Graph U-Nets. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44 (2019), 4948–4960.

[21] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural Message Passing for Quantum Chemistry. *ArXiv* abs/1704.01212 (2017).

[22] Zhichun Guo, Chuxu Zhang, W. Yu, John E. Herr, O. Wiest, Meng Jiang, and N. Chawla. 2021. Few-Shot Graph Learning for Molecular Property Prediction. *Proceedings of the Web Conference 2021* (2021).

[23] Arman Hasanzadeh, Ehsan Hajiramezanali, Shahin Boluki, Mingyuan Zhou, Nick Duffield, Krishna Narayanan, and Xiaoning Qian. 2020. Bayesian graph neural networks with adaptive connection sampling. In *International conference on machine learning*. PMLR, 4094–4104.

[24] Adrián Javaloy, Pablo Sánchez-Martín, Amit Levi, and Isabel Valera. 2022. Learnable Graph Convolutional Attention Networks. *ArXiv* abs/2211.11853 (2022).

[25] Thomas Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. *ArXiv* abs/1609.02907 (2016). https://api.semanticscholar.org/CorpusID:3144218

[26] Boris Knyazev, Graham W. Taylor, and Mohamed R. Amer. 2019. Understanding Attention and Generalization in Graph Neural Networks. In *Neural Information Processing Systems*. https://api.semanticscholar.org/CorpusID:195069083

[27] Volodymyr Kuleshov and Doina Precup. 2014. Algorithms for multi-armed bandit problems. *arXiv preprint arXiv:1402.6028* (2014).

[28] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. 2019. Self-Attention Graph Pooling. *ArXiv* abs/1904.08082 (2019). https://api.semanticscholar.org/CorpusID:119314157

[29] Jiayu Li, Tianyun Zhang, Hao Tian, Shengmin Jin, Makan Fardad, and Reza Zafarani. 2020. Sgcn: A graph sparsifier based on graph convolutional networks. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 275–287.

[30] Andreas Loukas. 2019. What graph neural networks cannot learn: depth vs width. *arXiv preprint arXiv:1907.03199* (2019).

[31] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. 2020. Parameterized explainer for graph neural network. *Advances in neural information processing systems* 33 (2020), 19620–19631.

[32] Dongsheng Luo, Wei Cheng, Wenchao Yu, Bo Zong, Jingchao Ni, Haifeng Chen, and Xiang Zhang. 2021. Learning to drop: Robust graph neural network via topological denoising. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 779–787.

[33] Nina Mazyavkina, S. V. Sviridov, Sergei Ivanov, and Evgeny Burnaev. 2020. Reinforcement Learning for Combinatorial Optimization: A Survey. *Comput. Oper. Res.* 134 (2020), 105400. https://api.semanticscholar.org/CorpusID:212633747

[34] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Kirkeby Fidjeland, Georg Ostrovski, Stig Petersen, Charlie Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518 (2015), 529–533.

[35] Mingshuo Nie, Dongming Chen, and Dongqi Wang. 2022. Reinforcement learning on graph: A survey. *ArXiv* abs/2204.06127 (2022).

[36] Danilo Numeroso and Davide Bacciu. 2021. MEG: Generating Molecular Counterfactual Explanations for Deep Graph Networks. *arXiv preprint arXiv:2104.08060* (2021).

[37] Kenta Oono and Taiji Suzuki. 2019. Graph neural networks exponentially lose expressive power for node classification. *arXiv preprint arXiv:1905.10947* (2019).

[38] Ricardo Ramirez, Yu-Chiao Chiu, Allen Hererra, Milad Mostavi, Joshua Ramirez, Yidong Chen, Yufei Huang, and Yu-Fang Jin. 2020. Classification of Cancer Types Using Graph Convolutional Neural Networks. In *Frontiers of Physics*.

[39] Yaniv Romano, Matteo Sesia, and Emmanuel J. Candès. 2020. Classification with Valid and Adaptive Coverage. *arXiv: Methodology* (2020). https://api.semanticscholar.org/CorpusID:219303493

[40] Yu Rong, Wen bing Huang, Tingyang Xu, and Junzhou Huang. 2019. DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. In *International Conference on Learning Representations*.

[41] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The Graph Neural Network Model. *IEEE Transactions on Neural Networks* 20 (2009), 61–80.

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

[43] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[44] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *ArXiv* abs/1707.06347 (2017).

[45] Oleksandr Shchur and Stephan Günnemann. 2018. Overlapping Community Detection with Graph Neural Networks. *ArXiv* abs/1909.12201 (2018).

[46] Qingyun Sun, Jianxin Li, Hao Peng, Jia Wu, Yuanxing Ning, Philip S. Yu, and Lifang He. 2021. SUGAR: Subgraph Neural Network with Reinforcement Pooling and Self-Supervised Mutual Information Mechanism. In *Proceedings of the Web Conference 2021* (Ljubljana, Slovenia) *(WWW '21)*. Association for Computing Machinery, New York, NY, USA, 2081–2091. https://doi.org/10.1145/3442381.3449822

[47] Richard S Sutton. 1992. Introduction: The challenge of reinforcement learning. *Reinforcement learning* (1992), 1–3.

[48] Richard S. Sutton and Andrew G. Barto. 2005. Reinforcement Learning: An Introduction. *IEEE Transactions on Neural Networks* 16 (2005), 285–286.

[49] Richard S. Sutton, David A. McAllester, Satinder Singh, and Y. Mansour. 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *NIPS*.

[50] Hannu (TT) Toivonen, Ashwin Srinivasan, Ross D. King, Stefan Kramer, and Christoph Helma. 2003. Statistical Evaluation of the Predictive Toxicology Challenge 2000-2001. *Bioinformatics* 19 10 (2003), 1183–93.

[51] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio', and Yoshua Bengio. 2017. Graph Attention Networks. *ArXiv* abs/1710.10903 (2017).

[52] Nikil Wale, Ian A. Watson, and George Karypis. 2006. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems* 14 (2006), 347–375.

[53] Runzhong Wang, Zhigang Hua, Gan Liu, Jiayi Zhang, Junchi Yan, Feng Qi, Shuang Yang, Jun Zhou, and Xiaokang Yang. 2021. A Bi-Level Framework for Learning to Solve Combinatorial Optimization on Graphs. *arXiv preprint arXiv:2106.04927* (2021).

[54] Ryan Wickman, Xiaofei Zhang, and Weizi Li. 2021. SparRL: Graph Sparsification via Deep Reinforcement Learning. *arXiv preprint arXiv:2112.01565* (2021).

[55] Shiwen Wu, Wentao Zhang, Fei Sun, and Bin Cui. 2020. Graph Neural Networks in Recommender Systems: A Survey. *Comput. Surveys* 55 (2020), 1 – 37.

[56] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2019. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems* 32 (2019), 4–24.

[57] Zhaoping Xiong, Dingyan Wang, Xiaohong Liu, Feisheng Zhong, Xiaozhe Wan, Xutong Li, Zhaojun Li, Xiaomin Luo, Kaixian Chen, Hualiang Jiang, and Mingyue Zheng. 2020. Pushing the boundaries of molecular representation for drug discovery with graph attention mechanism. *Journal of medicinal chemistry* (2020).

[58] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How Powerful are Graph Neural Networks? *ArXiv* abs/1810.00826 (2018).

[59] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. 2018. Hierarchical Graph Representation Learning with Differentiable Pooling. In *Neural Information Processing Systems*.

[60] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. GNNexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems* 32 (2019).

[61] Zhaoning Yu and Hongyang Gao. 2022. MotifExplainer: a Motif-based Graph Neural Network Explainer. *arXiv preprint arXiv:2202.00519* (2022).

[62] Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. 2020. Xgnn: Towards model-level explanations of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 430–438.

[63] Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. 2021. On Explainability of Graph Neural Networks via Subgraph Explorations. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*. 12241–12252.

[64] Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, and Wei Wang. 2020. Robust graph representation learning via neural sparsification. In *International Conference on Machine Learning*. PMLR, 11458–11468.

[65] Liyuan Zheng, Tanner Fiez, Zane Alumbaugh, Benjamin J. Chasnov, and Lillian J. Ratliff. 2021. Stackelberg Actor-Critic: Game-Theoretic Reinforcement Learning Algorithms. *ArXiv* abs/2109.12286 (2021).

Table 4. **Best configuration for the Vanilla GNN.** Split sizes (#0), Batch size (#1), Dropout rate (#2), Batch normalizing (#3), Dimension of hidden layers (#4), Number of GNN layers (#5), Global pooling type (#6), Classifier scheduler factor (#7), Classifier learning rate (#8), ϵ (#9), Trainable ϵ (#10), Number of heads (#11).

Param	Archi	Datasets								
		BZR	**COX2**	**DD**	**ENZYMES**	**MUTAG**	**NCI1**	**NCI109**	**PROTEINS**	**PTC**
#0	**GIN**	[0.6, 0.3, 0.1]	[0.4, 0.5, 0.1]	[0.6, 0.3, 0.1]	[0.5, 0.4, 0.1]	[0.4, 0.5, 0.1]	[0.5, 0.4, 0.1]	[0.4, 0.5, 0.1]	[0.4, 0.5, 0.1]	[0.6, 0.3, 0.1]
	GAT	[0.6, 0.3, 0.1]	[0.5, 0.4, 0.1]	[0.5, 0.4, 0.1]	[0.5, 0.4, 0.1]	[0.5, 0.4, 0.1]	[0.6, 0.3, 0.1]	[0.6, 0.3, 0.1]	[0.4, 0.5, 0.1]	[0.5, 0.4, 0.1]
	GCN	[0.6, 0.3, 0.1]	[0.5, 0.4, 0.1]	[0.6, 0.3, 0.1]	[0.5, 0.4, 0.1]	[0.5, 0.4, 0.1]	[0.6, 0.3, 0.1]	[0.6, 0.3, 0.1]	[0.4, 0.5, 0.1]	[0.4, 0.5, 0.1]
#1	**GIN**	16	16	32	16	16	16	32	16	32
	GAT	32	16	32	16	16	32	32	16	16
	GCN	16	16	32	16	32	32	16	32	32
#2	**GIN**	0.0	0.2	0.5	0.2	0.0	0.0	0.1	0.1	0.4
	GAT	0.1	0.2	0.3	0.4	0.3	0.2	0.1	0.1	0.5
	GCN	0.3	0.1	0.3	0.0	0.4	0.0	0.1	0.4	0.5
#3	**GIN**	True	False	True	True	True	True	True	False	False
	GAT	True	True	False	True	True	True	False	False	True
	GCN	True	False	False	True	False	True	True	False	False
#4	**GIN**	128	64	32	64	16	128	128	128	16
	GAT	16	128	16	128	32	128	128	16	16
	GCN	64	128	32	128	16	64	128	32	32
#5	**GIN**	3	1	1	1	3	1	3	1	3
	GAT	1	2	1	4	4	4	3	2	4
	GCN	1	4	3	3	4	3	4	1	1
#6	**GIN**	['mean', 'add']	['add']	['mean', 'add']	['mean']	['mean', 'add']	['mean']	['mean']	['add']	['mean']
	GAT	['add']	['mean']	['mean', 'add']	['mean']	['mean', 'add']	['add']	['mean', 'add']	['add']	['mean', 'add']
	GCN	['mean']	['mean', 'add']	['mean', 'add']	['mean']	['mean', 'add']	['mean', 'add']	['add']	['add']	['mean']
#7	**GIN**	0.95	0.99	0.99	0.9	0.95	0.9	0.99	0.9	0.99
	GAT	0.95	0.95	0.99	0.95	0.99	0.95	0.99	0.95	0.9
	GCN	0.99	0.99	0.99	0.95	0.99	0.99	0.9	0.99	0.99
#8	**GIN**	0.0001	0.001	0.005	0.0001	0.001	0.01	0.001	0.0005	0.005
	GAT	0.01	0.0001	0.0001	0.001	0.005	0.005	0.001	0.005	0.01
	GCN	0.01	0.0001	0.001	0.01	0.005	0.005	0.001	0.001	0.01
#9	**GIN**	0.3	0.2	0.3	0.0	0.2	0.2	0.2	0.0	0.2
	GAT	-	-	-	-	-	-	-	-	-
	GCN	-	-	-	-	-	-	-	-	-
#10	**GIN**	False	True	True	True	True	False	True	False	False
	GAT	-	-	-	-	-	-	-	-	-
	GCN	-	-	-	-	-	-	-	-	-
#11	**GIN**	-	-	-	-	-	-	-	-	-
	GAT	2.0	4.0	4.0	2.0	2.0	4.0	4.0	2.0	4.0
	GCN	-	-	-	-	-	-	-	-	-

A TRAINING DETAILS

In this section, we provide further details of the experimental setup used to obtain our results. For all experiments, we train the models using a fixed seed of 0, and the reported results represent the mean and standard deviation over five different dataset splits. We run the models up to 1000 epochs, doing early stopping if the accuracy does not improve for 500 epochs. All experiments were conducted on a single CPU with 8GB of RAM. We carry out hyperparameter tuning using random sampling. We explored 20 different configurations for each model, data, and GNN architecture. Complete details regarding the cross-validated hyperparameters can be found in our GitHub repository at https://github.com/XXXX/XXXX. Table 4 presents the best configurations achieved for each dataset and architecture for the Vanilla GNN baselines. These configurations were subsequently used for training CORES. The best configuration for CORES_{N} is detailed in Table 7, while for CORES_{E}, it can be found in Table 8. The best configurations for SAGPooling and GPool are outlined in Table 5 and Table 6, respectively.

Table 5. **Best configuration for SAGPooling.** Split sizes (#0), Batch size (#1), Early stopping clf. patience (#2), Dropout rate (#3), Batch normalizing (#4), Dimension of hidden layers (#5), Number of GNN layers (#6), Global pooling type (#7), Classifier scheduler factor (#8), Classifier learning rate (#9), TopK multiplier (#10), TopK ratio (#11), Number of heads (#12), ϵ (#13), Trainble ϵ (#14)

Param	Archi	Datasets								
		BZR	**COX2**	**DD**	**ENZYMES**	**MUTAG**	**NCI1**	**NCI109**	**PROTEINS**	**PTC**
#0	**GCN**	[0.5, 0.4, 0.1]	[0.5, 0.4, 0.1]	[0.4, 0.5, 0.1]	[0.5, 0.4, 0.1]	[0.5, 0.4, 0.1]	[0.5, 0.4, 0.1]	[0.5, 0.4, 0.1]	[0.4, 0.5, 0.1]	[0.6, 0.3, 0.1]
	GAT	[0.4, 0.5, 0.1]	[0.4, 0.5, 0.1]	[0.4, 0.5, 0.1]	[0.5, 0.4, 0.1]	[0.4, 0.5, 0.1]	[0.6, 0.3, 0.1]	[0.5, 0.4, 0.1]	[0.4, 0.5, 0.1]	[0.4, 0.5, 0.1]
	GIN	[0.5, 0.4, 0.1]	[0.5, 0.4, 0.1]	[0.5, 0.4, 0.1]	[0.6, 0.3, 0.1]	[0.5, 0.4, 0.1]	[0.5, 0.4, 0.1]	[0.6, 0.3, 0.1]	[0.6, 0.3, 0.1]	[0.5, 0.4, 0.1]
#1	**GCN**	16	16	16	16	32	16	16	16	32
	GAT	16	16	16	16	16	16	32	16	16
	GIN	16	16	16	32	32	16	32	32	16
#2	**GCN**	800	650	500	650	500	650	650	500	650
	GAT	800	800	500	800	800	650	500	800	800
	GIN	800	500	650	500	800	800	500	500	650
#3	**GCN**	0.4	0.0	0.5	0.1	0.4	0.0	0.1	0.5	0.4
	GAT	0.3	0.3	0.1	0.1	0.4	0.1	0.4	0.3	0.3
	GIN	0.2	0.3	0.5	0.0	0.4	0.2	0.0	0.0	0.5
#4	**GCN**	True	False	False	True	True	False	True	False	False
	GAT	False	False	False	False	True	True	False	False	False
	GIN	True	False	False	True	False	True	True	True	False
#5	**GCN**	128	32	32	128	32	32	128	32	64
	GAT	16	16	64	128	16	32	16	16	16
	GIN	32	128	128	64	16	32	64	64	128
#6	**GCN**	4	1	4	4	1	1	4	4	3
	GAT	1	1	2	2	2	3	2	1	1
	GIN	4	4	3	1	2	4	1	1	3
#7	**GCN**	['add']	['mean', 'add']	['add']	['mean']	['mean', 'add']	['mean', 'add']	['mean']	['add']	['mean']
	GAT	['mean', 'add']	['mean', 'add']	['mean', 'add']	['mean', 'add']	['mean', 'add']	['mean']	['mean', 'add']	['mean', 'add']	['mean', 'add']
	GIN	['mean']	['mean']	['mean']	['add']	['mean', 'add']	['mean']	['add']	['add']	['mean']
#8	**GCN**	0.99	0.95	0.99	0.95	0.9	0.95	0.95	0.99	0.9
	GAT	0.9	0.9	0.9	0.99	0.99	0.95	0.99	0.9	0.9
	GIN	0.9	0.99	0.95	0.95	0.99	0.9	0.95	0.95	0.95
#9	**GCN**	0.01	0.005	0.0005	0.0001	0.01	0.005	0.0001	0.0005	0.0001
	GAT	0.01	0.01	0.01	0.005	0.005	0.0001	0.005	0.01	0.01
	GIN	0.001	0.0005	0.0005	0.001	0.005	0.001	0.001	0.001	0.0005
#10	**GCN**	0.5	2.0	0.5	1.5	1.0	2.0	1.5	0.5	0.5
	GAT	1.0	1.0	2.0	1.0	1.5	1.5	0.5	1.0	1.0
	GIN	1.5	2.0	0.5	0.5	0.5	1.5	0.5	0.5	0.5
#11	**GCN**	0.95	0.95	0.9	0.3	0.3	0.95	0.3	0.9	0.2
	GAT	0.9	0.9	0.5	0.9	0.2	0.6	0.8	0.9	0.9
	GIN	0.3	0.4	0.8	0.3	0.8	0.3	0.3	0.3	0.8
#12	**GCN**	-	-	-	-	-	-	-	-	-
	GAT	1.0	1.0	1.0	1.0	1.0	1.0	4.0	1.0	1.0
	GIN	-	-	-	-	-	-	-	-	-
#13	**GCN**	-	-	-	-	-	-	-	-	-
	GAT	-	-	-	-	-	-	-	-	-
	GIN	0.4	0.2	0.4	0.4	0.3	0.4	0.4	0.4	0.4
#14	**GCN**	-	-	-	-	-	-	-	-	-
	GAT	-	-	-	-	-	-	-	-	-
	GIN	False	True	True	True	False	False	True	True	True

Table 6. **Best configuration for the GPool.** Split sizes (#0), Batch size (#1), Early stopping clf. patience (#2), Dropout rate (#3), Batch normalizing (#4), Dimension of hidden layers (#5), Number of GNN layers (#6), Global pooling type (#7), Classifier scheduler factor (#8), Classifier learning rate (#9), TopK multiplier (#10), TopK ratio (#11), Number of heads (#12), ϵ (#13), Trainble ϵ (#14).

Param	Archi	Datasets								
		BZR	**COX2**	**DD**	**ENZYMES**	**MUTAG**	**NCI1**	**NCI109**	**PROTEINS**	**PTC**
#0	**GAT**	[0.5, 0.4, 0.1]	[0.6, 0.3, 0.1]	[0.6, 0.3, 0.1]	[0.4, 0.5, 0.1]	[0.4, 0.5, 0.1]	[0.4, 0.5, 0.1]	[0.4, 0.5, 0.1]	[0.4, 0.5, 0.1]	[0.4, 0.5, 0.1]
	GIN	[0.4, 0.5, 0.1]	[0.4, 0.5, 0.1]	[0.4, 0.5, 0.1]	[0.4, 0.5, 0.1]	[0.5, 0.4, 0.1]	[0.4, 0.5, 0.1]	[0.4, 0.5, 0.1]	[0.4, 0.5, 0.1]	[0.5, 0.4, 0.1]
	GCN	[0.4, 0.5, 0.1]	[0.4, 0.5, 0.1]	[0.4, 0.5, 0.1]	[0.4, 0.5, 0.1]	[0.5, 0.4, 0.1]	[0.4, 0.5, 0.1]	[0.4, 0.5, 0.1]	[0.4, 0.5, 0.1]	[0.6, 0.3, 0.1]
#1	**GAT**	16	16	16	32	32	32	32	32	32
	GIN	32	16	16	32	16	32	32	16	32
	GCN	16	16	32	32	16	32	32	32	16
#2	**GAT**	500	650	800	500	500	500	500	650	800
	GIN	500	800	800	500	800	500	500	800	800
	GCN	650	650	500	500	650	500	500	800	650
#3	**GAT**	0.0	0.3	0.5	0.1	0.1	0.1	0.1	0.1	0.3
	GIN	0.1	0.0	0.5	0.1	0.4	0.1	0.1	0.5	0.2
	GCN	0.0	0.0	0.4	0.1	0.0	0.3	0.1	0.5	0.1
#4	**GAT**	True	True	False	True	True	True	True	False	False
	GIN	True	True	False	True	False	True	True	False	False
	GCN	True	True	False	True	True	True	True	True	False
#5	**GAT**	16	64	16	64	64	64	64	16	64
	GIN	64	32	16	64	32	64	64	16	128
	GCN	128	128	128	64	32	128	64	16	16
#6	**GAT**	2	4	1	2	2	2	2	1	3
	GIN	2	1	1	2	1	2	2	1	4
	GCN	2	2	2	2	4	3	2	3	3
#7	**GAT**	['mean', 'add']	['mean']	['mean', 'add']	['mean']	['mean']	['mean']	['mean']	['add']	['add']
	GIN	['mean']	['mean']	['mean', 'add']	['mean']	['mean', 'add']	['mean']	['mean']	['mean', 'add']	['add']
	GCN	['mean']	['mean']	['add']	['mean']	['mean', 'add']	['mean', 'add']	['mean']	['mean', 'add']	['mean']
#8	**GAT**	0.9	0.95	0.9	0.9	0.9	0.9	0.9	0.95	0.99
	GIN	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
	GCN	0.95	0.95	0.99	0.9	0.99	0.9	0.9	0.99	0.95
#9	**GAT**	0.005	0.0005	0.001	0.01	0.01	0.01	0.01	0.001	0.0005
	GIN	0.01	0.0001	0.001	0.01	0.01	0.01	0.01	0.001	0.0001
	GCN	0.0001	0.0001	0.01	0.01	0.01	0.001	0.01	0.001	0.005
#10	**GAT**	2.0	0.5	1.5	2.0	2.0	2.0	2.0	1.5	0.5
	GIN	2.0	2.0	1.5	2.0	1.0	2.0	2.0	1.5	0.5
	GCN	1.0	1.0	1.5	2.0	0.5	1.0	2.0	0.5	1.5
#11	**GAT**	0.4	0.5	0.4	0.95	0.95	0.95	0.95	0.5	0.5
	GIN	0.95	0.9	0.4	0.95	0.5	0.95	0.95	0.4	0.4
	GCN	0.95	0.95	0.4	0.95	0.7	0.95	0.95	0.5	0.8
#12	**GAT**	4.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	GIN	-	-	-	-	-	-	-	-	-
	GCN	-	-	-	-	-	-	-	-	-
#13	**GAT**	-	-	-	-	-	-	-	-	-
	GIN	0.0	0.4	0.1	0.0	0.1	0.0	0.0	0.1	0.2
	GCN	-	-	-	-	-	-	-	-	-
#14	**GAT**	-	-	-	-	-	-	-	-	-
	GIN	True	True	False	True	True	True	True	False	True
	GCN	-	-	-	-	-	-	-	-	-

Table 7. **Best configuration for CORES$_{\mathbf{N}}$.** Early stopping PPO patience (#0), Number of environment steps (#1), Number of PPO epochs (#2), Environment penalty size (#3), RL scheduler factor (#4), Ratio of the critic learning rate (#5), PPO entropy coefficient (#6), PPO MSE coefficient (#7), PPO clip value ϵ (#8), Conformal error rate α (#9), d (#10), λ (#11).

Param	Archi	Datasets								
		BZR	**COX2**	**DD**	**ENZYMES**	**MUTAG**	**NCI1**	**NCI109**	**PROTEINS**	**PTC**
#0	**GIN**	15	5	5	10	10	5	15	5	5
	GCN	10	5	5	15	5	15	10	15	15
	GAT	15	5	5	10	15	15	10	10	5
#1	**GIN**	128	128	128	128	128	128	64	64	64
	GCN	32	64	64	128	64	32	32	32	128
	GAT	64	32	32	32	128	64	32	32	64
#2	**GIN**	10	15	15	5	15	15	10	5	15
	GCN	10	15	15	5	15	3	10	5	5
	GAT	10	15	15	10	5	10	10	5	5
#3	**GIN**	0.5	0.5	0.5	0.5	0.5	0.5	1.5	1.5	1.0
	GCN	0.5	0.5	0.5	1.0	0.5	0.5	0.5	1.5	1.5
	GAT	1.5	0.5	1.5	0.5	1.5	1.5	0.5	1.0	1.5
#4	**GIN**	0.99	0.95	0.95	0.99	0.9	0.95	0.9	0.9	0.95
	GCN	0.99	0.95	0.95	0.99	0.95	0.99	0.99	0.9	0.9
	GAT	0.9	0.95	0.99	0.99	0.9	0.9	0.99	0.95	0.9
#5	**GIN**	3.0	2.5	2.5	1.0	3.0	2.5	2.5	1.0	3.0
	GCN	2.5	1.5	1.5	2.0	1.5	3.0	2.5	2.5	1.5
	GAT	2.5	2.0	2.5	2.5	1.5	2.5	2.5	1.0	1.0
#6	**GIN**	0.001	0.0001	0.0001	0.01	0.001	0.0001	0.0001	0.001	0.0001
	GCN	0.0001	0.01	0.01	0.0001	0.01	0.01	0.0001	0.01	0.001
	GAT	0.0001	0.001	0.001	0.0001	0.001	0.0001	0.0001	0.001	0.001
#7	**GIN**	0.5	5.0	5.0	2.0	1.0	5.0	2.0	3.0	0.5
	GCN	1.0	0.5	0.5	0.1	0.5	1.0	1.0	5.0	3.0
	GAT	2.0	1.0	3.0	1.0	3.0	2.0	1.0	0.1	3.0
#8	**GIN**	0.2	0.4	0.4	0.1	0.2	0.4	0.2	0.1	0.4
	GCN	0.4	0.3	0.3	0.3	0.3	0.1	0.4	0.3	0.1
	GAT	0.2	0.2	0.4	0.4	0.1	0.2	0.4	0.4	0.1
#9	**GIN**	0.2	0.05	0.05	0.1	0.2	0.05	0.2	0.05	0.05
	GCN	0.05	0.2	0.2	0.05	0.2	0.1	0.05	0.2	0.05
	GAT	0.2	0.2	0.2	0.05	0.05	0.2	0.05	0.05	0.05
#10	**GIN**	0.2	0.3	0.3	0.7	0.7	0.3	0.4	0.2	0.5
	GCN	0.9	0.8	0.8	0.6	0.8	0.4	0.9	0.5	0.2
	GAT	0.4	0.6	0.6	0.9	0.2	0.4	0.9	0.6	0.2
#11	**GIN**	0.2	1.0	1.0	0.9	0.1	1.0	0.4	0.0	0.4
	GCN	0.2	0.7	0.7	1.0	0.7	0.8	0.2	0.8	0.7
	GAT	0.4	0.3	0.8	0.2	0.7	0.4	0.2	0.5	0.0

Table 8. **Best configuration for CORES$_{\mathbf{E}}$.** Early stopping PPO patience (#0), Number of environment steps (#1), Number of PPO epochs (#2), Environment penalty size (#3), RL scheduler factor (#4), Ratio of the critic learning rate (#5), PPO entropy coefficient (#6), PPO MSE coefficient (#7), PPO clip value ϵ (#8), Conformal error rate α (#9), d (#10), λ (#11).

Param	Archi	Datasets								
		BZR	**COX2**	**DD**	**ENZYMES**	**MUTAG**	**NCI1**	**NCI109**	**PROTEINS**	**PTC**
	GIN	15	5	5	10	10	5	15	5	5
#0	**GCN**	10	5	5	15	5	15	10	15	15
	GAT	15	5	5	10	15	15	10	10	5
	GIN	128	128	128	128	128	128	64	64	64
#1	**GCN**	32	64	64	128	64	32	32	32	128
	GAT	64	32	32	32	128	64	32	32	64
	GIN	10	15	15	5	15	15	10	5	15
#2	**GCN**	10	15	15	5	15	3	10	5	5
	GAT	10	15	15	10	5	10	10	5	5
	GIN	0.5	0.5	0.5	0.5	0.5	0.5	1.5	1.5	1.0
#3	**GCN**	0.5	0.5	0.5	1.0	0.5	0.5	0.5	1.5	1.5
	GAT	1.5	0.5	1.5	0.5	1.5	1.5	0.5	1.0	1.5
	GIN	0.99	0.95	0.95	0.99	0.9	0.95	0.9	0.9	0.95
#4	**GCN**	0.99	0.95	0.95	0.99	0.95	0.99	0.99	0.9	0.9
	GAT	0.9	0.95	0.99	0.99	0.9	0.9	0.99	0.95	0.9
	GIN	3.0	2.5	2.5	1.0	3.0	2.5	2.5	1.0	3.0
#5	**GCN**	2.5	1.5	1.5	2.0	1.5	3.0	2.5	2.5	1.5
	GAT	2.5	2.0	2.5	2.5	1.5	2.5	2.5	1.0	1.0
	GIN	0.001	0.0001	0.0001	0.01	0.001	0.0001	0.0001	0.001	0.0001
#6	**GCN**	0.0001	0.01	0.01	0.0001	0.01	0.01	0.0001	0.01	0.001
	GAT	0.0001	0.001	0.001	0.0001	0.001	0.0001	0.0001	0.001	0.001
	GIN	0.5	5.0	5.0	2.0	1.0	5.0	2.0	3.0	0.5
#7	**GCN**	1.0	0.5	0.5	0.1	0.5	1.0	1.0	5.0	3.0
	GAT	2.0	1.0	3.0	1.0	3.0	2.0	1.0	0.1	3.0
	GIN	0.2	0.4	0.4	0.1	0.2	0.4	0.2	0.1	0.4
#8	**GCN**	0.4	0.3	0.3	0.3	0.3	0.1	0.4	0.3	0.1
	GAT	0.2	0.2	0.4	0.4	0.1	0.2	0.4	0.4	0.1
	GIN	0.2	0.05	0.05	0.1	0.2	0.05	0.2	0.05	0.05
#9	**GCN**	0.05	0.2	0.2	0.05	0.2	0.1	0.05	0.2	0.05
	GAT	0.2	0.2	0.2	0.05	0.05	0.2	0.05	0.05	0.05
	GIN	0.2	0.3	0.3	0.7	0.7	0.3	0.4	0.2	0.5
#10	**GCN**	0.9	0.8	0.8	0.6	0.8	0.4	0.9	0.5	0.2
	GAT	0.4	0.6	0.6	0.9	0.2	0.4	0.9	0.6	0.2
	GIN	0.2	1.0	1.0	0.9	0.1	1.0	0.4	0.0	0.4
#11	**GCN**	0.2	0.7	0.7	1.0	0.7	0.8	0.2	0.8	0.7
	GAT	0.4	0.3	0.8	0.2	0.7	0.4	0.2	0.5	0.0

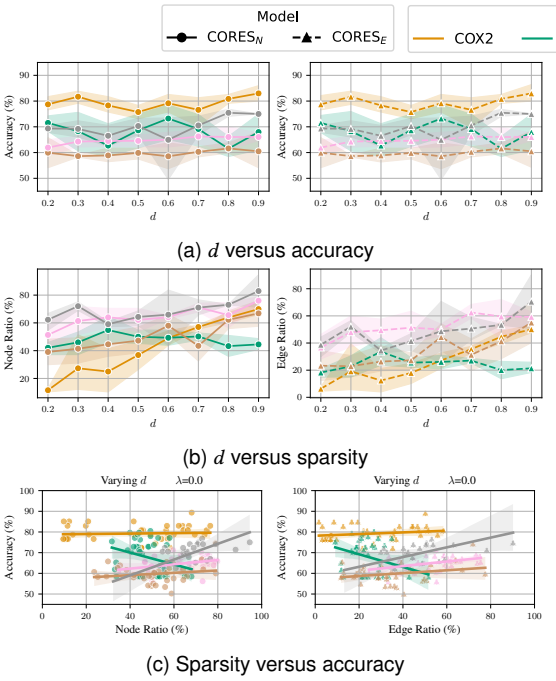


Fig. 6. Ablation study on the maximum desired ratio d for 5 different runs for the GIN architecture.

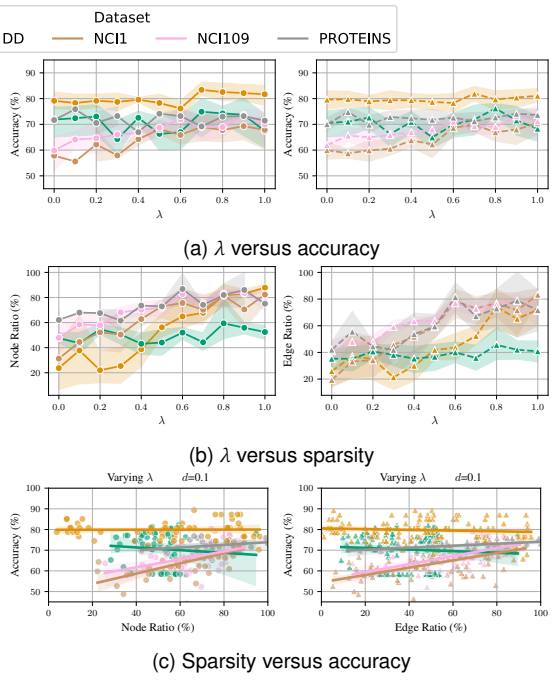


Fig. 7. Ablation study on λ for 5 different runs for the GIN architecture.

B EXTRA RESULTS IMPACT OF HYPERPARAMETERS

In this section, we present an ablation study on the impact of two key hyperparameters of CORES, namely d and λ, for the datasets not included in the main manuscript: COX2, DD, NCI1, NCI109, and PROTEINS. Figure 6 illustrates the results of the ablation study for parameter d, while the results for parameter λ are shown in Figure 7.

The findings from this ablation study are consistent with those reported in the main manuscript. In the top row figures, we observe that the accuracy remains relatively stable when varying either d or λ, improving only as the values increase for certain datasets, such as *NCI*1 or *NCI*109. In the middle row figures, we find a positive correlation between both parameters and the node/edge ratio for most of the datasets, and no correlation for others. Remarkably, the correlation is never negative. Finally, the bottom row figures reveal a positive correlation between the node/edge ratio and accuracy for the majority of the datasets. This pattern holds except for the DD dataset, which present a negative correlation. Interestingly this dataset presents the most significant disparities in terms of the number of nodes and edges compared to the other datasets (see Table 1).

Table 9. **Model comparison results with GAT architecture.** We show the mean over 5 independent runs and the standard deviation as the subindex. All metrics are shown in percentage. The last rows include the average ranking of the model across datasets.

Dataset	Metric	Models				
		GNN	**SAGPool**	**GPool**	**CORES$_N$**	**CORES$_E$**
BZR	**Accuracy**	$80.98_{4.36}$	$83.90_{3.27}$	$79.51_{6.12}$	$83.90_{4.43}$	$81.46_{4.08}$
	Node Ratio	-	-	$41.11_{0.11}$	$37.99_{10.27}$	$100.00_{0.00}$
	Edge Ratio	-	-	$20.93_{0.70}$	$29.92_{8.83}$	$57.26_{21.50}$
COX2	**Accuracy**	$82.13_{4.15}$	$80.00_{4.41}$	$77.87_{3.87}$	$81.70_{4.41}$	$80.85_{2.61}$
	Node Ratio	-	-	$50.62_{0.09}$	$77.91_{2.29}$	$100.00_{0.00}$
	Edge Ratio	-	-	$20.75_{0.51}$	$62.62_{3.24}$	$66.18_{0.38}$
DD	**Accuracy**	$76.10_{3.02}$	$75.76_{3.72}$	$77.80_{1.84}$	$75.42_{4.19}$	$76.44_{2.19}$
	Node Ratio	-	-	$40.24_{0.03}$	$75.16_{1.39}$	$100.00_{0.00}$
	Edge Ratio	-	-	$15.63_{0.29}$	$56.42_{2.16}$	$58.62_{2.51}$
ENZYMES	**Accuracy**	$71.33_{10.63}$	$45.00_{7.64}$	$63.93_{6.35}$	$69.84_{4.43}$	$67.54_{12.23}$
	Node Ratio	-	-	$96.67_{0.14}$	$80.26_{1.03}$	$100.00_{0.00}$
	Edge Ratio	-	-	$93.27_{0.23}$	$65.92_{1.80}$	$61.60_{9.06}$
MUTAG	**Accuracy**	$81.43_{6.39}$	$88.57_{3.91}$	$78.57_{7.14}$	$80.00_{5.98}$	$81.43_{6.02}$
	Node Ratio	-	-	$97.81_{0.36}$	$75.72_{4.04}$	$100.00_{0.00}$
	Edge Ratio	-	-	$95.65_{2.50}$	$58.74_{6.05}$	$55.25_{14.77}$
NCI1	**Accuracy**	$79.42_{1.34}$	$68.93_{1.37}$	$72.43_{2.11}$	$73.43_{2.67}$	$75.38_{1.22}$
	Node Ratio	-	-	$96.89_{0.08}$	$63.49_{3.58}$	$100.00_{0.00}$
	Edge Ratio	-	-	$94.07_{0.34}$	$53.33_{2.81}$	$88.68_{6.67}$
NCI109	**Accuracy**	$76.90_{1.34}$	$67.46_{4.31}$	$67.22_{3.25}$	$74.72_{2.25}$	$72.78_{2.29}$
	Node Ratio	-	-	$96.79_{0.04}$	$71.55_{11.30}$	$100.00_{0.00}$
	Edge Ratio	-	-	$94.70_{0.98}$	$62.37_{12.96}$	$61.37_{13.13}$
PROTEINS	**Accuracy**	$75.36_{3.00}$	$72.68_{3.26}$	$73.21_{3.22}$	$73.39_{3.86}$	$73.57_{1.62}$
	Node Ratio	-	-	$51.15_{0.17}$	$67.94_{2.97}$	$100.00_{0.00}$
	Edge Ratio	-	-	$28.39_{0.42}$	$46.56_{4.27}$	$52.82_{1.86}$
PTC	**Accuracy**	$64.00_{4.33}$	$65.00_{6.39}$	$62.44_{4.12}$	$62.96_{4.17}$	$59.43_{4.69}$
	Node Ratio	-	-	$52.46_{0.41}$	$45.61_{6.00}$	$100.00_{0.00}$
	Edge Ratio	-	-	$36.58_{3.53}$	$21.24_{6.53}$	$76.36_{14.75}$
Avg. Rank	**Accuracy**	1.89	3.33	3.89	3.00	2.89
	Node Ratio	-	-	1.67	1.33	-
	Edge Ratio	-	-	2.33	1.56	2.11

C EXTRA COMPARISON RESULTS

Here, we analyze the results of how switching the base model affects the performance. Table 10 and Table 9 summarizes the results for two base architectures GCN and GAT. It can be observed that even when the base architecture changes our models CORES$_{\text{N}}$ and CORES$_{\text{E}}$ achieve competitive accuracy compared to the respective baseline and achieve top rankings in terms of node/edge ratio.

Table 10. **Model comparison results with GCN architecture** We show the mean over 5 independent runs and the standard deviation as the subindex. All metrics are shown in percentage. The last rows include the average ranking of the model across datasets.

Dataset	Metric	Models				
		GNN	**SAGPool**	**GPool**	**CORES$_N$**	**CORES$_E$**
BZR	**Accuracy**	$86.83_{4.08}$	$83.41_{4.69}$	$84.39_{4.08}$	$82.44_{4.69}$	$80.98_{5.29}$
	Node Ratio	-	-	$96.46_{0.19}$	$76.98_{6.06}$	$100.00_{0.00}$
	Edge Ratio	-	-	$92.87_{0.58}$	$61.17_{11.31}$	$46.16_{5.68}$
COX2	**Accuracy**	$79.15_{4.85}$	$80.43_{3.81}$	$80.00_{1.90}$	$80.85_{3.01}$	$81.70_{4.90}$
	Node Ratio	-	-	$96.14_{0.18}$	$72.49_{4.24}$	$100.00_{0.00}$
	Edge Ratio	-	-	$92.67_{0.71}$	$53.19_{8.07}$	$31.27_{23.60}$
DD	**Accuracy**	$78.47_{3.47}$	$75.93_{4.39}$	$76.10_{5.06}$	$79.32_{2.58}$	$75.76_{4.05}$
	Node Ratio	-	-	$40.23_{0.03}$	$56.44_{2.50}$	$100.00_{0.00}$
	Edge Ratio	-	-	$16.48_{0.13}$	$32.01_{2.89}$	$51.39_{1.90}$
ENZYMES	**Accuracy**	$66.00_{5.35}$	$52.00_{5.70}$	$64.59_{4.86}$	$58.31_{8.47}$	$57.02_{11.62}$
	Node Ratio	-	-	$96.67_{0.14}$	$77.17_{7.28}$	$100.00_{0.00}$
	Edge Ratio	-	-	$93.39_{0.35}$	$62.33_{9.26}$	$74.20_{35.99}$
MUTAG	**Accuracy**	$81.43_{3.91}$	$80.00_{3.19}$	$78.57_{5.05}$	$81.43_{10.83}$	$87.14_{6.56}$
	Node Ratio	-	-	$73.13_{0.55}$	$79.94_{2.79}$	$100.00_{0.00}$
	Edge Ratio	-	-	$68.22_{2.47}$	$63.52_{4.23}$	$32.66_{8.79}$
NCI1	**Accuracy**	$77.33_{2.09}$	$73.58_{1.64}$	$72.14_{2.64}$	$66.45_{7.71}$	$65.94_{6.31}$
	Node Ratio	-	-	$96.89_{0.08}$	$76.47_{17.88}$	$100.00_{0.00}$
	Edge Ratio	-	-	$96.06_{0.20}$	$64.02_{18.84}$	$37.98_{14.34}$
NCI109	**Accuracy**	$79.32_{1.57}$	$71.19_{2.03}$	$71.53_{2.12}$	$76.08_{2.71}$	$77.14_{2.02}$
	Node Ratio	-	-	$96.79_{0.04}$	$78.78_{4.15}$	$100.00_{0.00}$
	Edge Ratio	-	-	$95.82_{0.11}$	$67.08_{4.42}$	$92.02_{1.93}$
PROTEINS	**Accuracy**	$72.86_{1.02}$	$72.86_{3.13}$	$75.18_{2.75}$	$73.21_{2.45}$	$75.89_{1.94}$
	Node Ratio	-	-	$51.15_{0.17}$	$74.66_{6.10}$	$100.00_{0.00}$
	Edge Ratio	-	-	$28.15_{0.67}$	$56.12_{9.77}$	$64.73_{0.98}$
PTC	**Accuracy**	$57.78_{7.71}$	$64.57_{3.83}$	$63.43_{1.28}$	$62.86_{2.86}$	$63.43_{3.13}$
	Node Ratio	-	-	$83.80_{0.74}$	$64.10_{4.19}$	$100.00_{0.00}$
	Edge Ratio	-	-	$78.21_{3.76}$	$40.21_{6.93}$	$44.47_{5.29}$
Avg. Rank	**Accuracy**	1.89	3.33	3.89	3.00	2.89
	Node Ratio	-	-	1.67	1.33	-
	Edge Ratio	-	-	2.56	1.67	1.78

Table 11. **Training time comparison.** We show the average training time along with the standard deviation of each model on each dataset for one epoch with a batch size of one. We use the GIN GNN architecture.

Dataset	Full Models		Sparse Models			
	Vanilla	DiffPool	SUGAR	GPool	CORES_N	CORES_E
BZR	2.64 $_{0.18}$	4.23 $_{0.12}$	-	3.44 $_{0.27}$	186.96 $_{61.70}$	121.94 $_{2.00}$
COX2	3.07 $_{0.15}$	4.83 $_{0.17}$	-	10.00 $_{2.70}$	411.93 $_{85.97}$	203.93 $_{1.86}$
DD	6.13 $_{0.13}$	-	-	33.31 $_{4.90}$	1848.21 $_{1.12}$	1857.59 $_{1.13}$
ENZYMES	2.40 $_{0.04}$	6.26 $_{0.31}$	132.86 $_{3.77}$	6.08 $_{0.30}$	233.78 $_{6.65}$	287.95 $_{2.20}$
MUTAG	1.09 $_{0.03}$	1.63 $_{0.04}$	47.82 $_{3.45}$	1.71 $_{0.19}$	24.97 $_{1.55}$	22.94 $_{0.30}$
NCI1	15.44 $_{0.19}$	41.74 $_{3.03}$	1143.85 $_{45.27}$	48.05 $_{8.12}$	1596.95 $_{107.83}$	1826.41 $_{51.06}$
NCI109	16.68 $_{1.15}$	43.73 $_{0.44}$	1195.10 $_{47.42}$	43.46 $_{3.09}$	1533.59 $_{15.51}$	1756.84 $_{37.12}$
PROTEINS	5.80 $_{0.23}$	11.79 $_{1.19}$	511.21 $_{22.63}$	13.58 $_{2.58}$	484.97 $_{14.74}$	505.89 $_{2.70}$
PTC	1.68 $_{0.16}$	3.00 $_{0.07}$	95.78 $_{1.47}$	3.28 $_{0.18}$	64.60 $_{1.23}$	67.19 $_{0.32}$

D TIME COMPLEXITY

We analyze the training and inference times of all models on each dataset. We run each model on every dataset for a single epoch, utilizing a batch size of one. To ensure statistical significance and reliability in our measurements, we repeat each experiment ten times. This repetition enables us to calculate the average training and inference times accurately. Such a meticulous approach guarantees robust and precise performance assessment for our models across the diverse set of datasets under examination.

Table 11 and Table 12 show the complete results for the training and inferences times, respectively. If we focus on Table 11, we observe the models that only rely on a single forward pass are two orders of magnitude faster than the models that rely on a reinforcement learning-based approach. Still, we observe that CORES achieves comparable speed as SUGAR, and sometimes it is faster, e.g., with PTC or MUTAG. In terms of inference time, in Table 12, we can observe that even though CORES is still one order of magnitude slower than the *Full Models*, it is considerably faster than SUGAR.

Table 12. **Inference time comparison.** We show the average inference time along with the standard deviation of each model on each dataset. We use the GIN GNN architecture.

Dataset	Full Models		Sparse Models			
	Vanilla	DiffPool	SUGAR	GPool	CORES_N	CORES_E
BZR	0.0025 $_{0.0001}$	0.1447 $_{0.0048}$	-	0.0045 $_{0.0003}$	0.0540 $_{0.0257}$	0.0425 $_{0.0011}$
COX2	0.0026 $_{0.0001}$	0.1550 $_{0.0093}$	-	0.0062 $_{0.0008}$	0.1031 $_{0.0349}$	0.0526 $_{0.0014}$
DD	0.0028 $_{0.0004}$	-	-	0.0085 $_{0.0006}$	0.2262 $_{0.0842}$	0.1206 $_{0.0016}$
ENZYMES	0.0024 $_{0.0000}$	0.1992 $_{0.0080}$	0.1399 $_{0.0094}$	0.0063 $_{0.0007}$	0.0496 $_{0.0014}$	0.0625 $_{0.0019}$
MUTAG	0.0029 $_{0.0001}$	0.0596 $_{0.0034}$	0.0135 $_{0.0036}$	0.0065 $_{0.0017}$	0.0448 $_{0.0056}$	0.0415 $_{0.0012}$
NCI1	0.0022 $_{0.0001}$	1.3756 $_{0.1246}$	1.0187 $_{0.1040}$	0.0058 $_{0.0004}$	0.0530 $_{0.0026}$	0.1301 $_{0.0602}$
NCI109	0.0023 $_{0.0001}$	1.3685 $_{0.0988}$	0.9477 $_{0.0132}$	0.0055 $_{0.0002}$	0.0500 $_{0.0006}$	0.0563 $_{0.0009}$
PROTEINS	0.0025 $_{0.0001}$	0.3835 $_{0.0619}$	3.9375 $_{0.0347}$	0.0058 $_{0.0007}$	0.0437 $_{0.0016}$	0.0439 $_{0.0008}$
PTC	0.0028 $_{0.0001}$	0.1076 $_{0.0050}$	0.0398 $_{0.0099}$	0.0058 $_{0.0005}$	0.0395 $_{0.0013}$	0.0418 $_{0.0005}$

Bibliography

[1] C. Agarwal, O. Queen, H. Lakkaraju, and M. Zitnik. "Evaluating explainability for graph neural networks". *Scientific Data* 10 (2022).

[2] S. Akansha. "Over-Squashing in Graph Neural Networks: A Comprehensive survey". *ArXiv* abs/2308.15568 (2023).

[3] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. "Optuna: A Next-generation Hyperparameter Optimization Framework". *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2019).

[4] S. Ali, T. Abuhmed, S. El-Sappagh, K. Muhammad, J. M. Alonso-Moral, R. Confalonieri, R. Guidotti, J. D. Ser, N. Díaz-Rodríguez, and F. Herrera. "Explainable Artificial Intelligence (XAI): What we know and what is left to attain Trustworthy Artificial Intelligence". *Inf. Fusion* 99 (2023), page 101805.

[5] U. Alon and E. Yahav. "On the Bottleneck of Graph Neural Networks and its Practical Implications". *ArXiv* abs/2006.05205 (2020).

[6] A. N. Angelopoulos and S. Bates. "A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification". *ArXiv* abs/2107.07511 (2021).

[7] A. N. Angelopoulos, S. Bates, J. Malik, and M. I. Jordan. "Uncertainty Sets for Image Classifiers using Conformal Prediction". *ArXiv* abs/2009.14193 (2020).

[8] A. Arnaiz-Rodríguez, A. Begga, F. Escolano, and N. Oliver. "DiffWire: Inductive Graph Rewiring via the Lovász Bound". *LOG IN*. 2022.

[9] L. Babai and L. Kucera. "Canonical labelling of graphs in linear average time". *Annual Symposium on Foundations of Computer Science* 20 (1979).

[10] A. Baranwal, K. Fountoulakis, and A. Jagannath. "Graph Convolution for Semi-Supervised Classification: Improved Linear Separability and Out-of-Distribution Generalization". *International Conference on Machine Learning (ICML)*. PMLR. 2021.

[11] A. Baranwal, K. Fountoulakis, and A. Jagannath. "Effects of Graph Convolutions in Deep Networks". *arXiv preprint arXiv:2204.09297* (2022).

[12] J. Bastings, I. Titov, W. Aziz, D. Marcheggiani, and K. Sima'an. "Graph convolutional encoders for syntax-aware neural machine translation". *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Volume 3. 2017.

[13] G. Bazhenov, D. Kuznedelev, A. Malinin, A. Babenko, and L. Prokhorenkova. "Evaluating Robustness and Uncertainty of Graph Models Under Structural Distributional Shifts". *Thirty-seventh Conference on Neural Information Processing Systems*. 2023.

[14] J. Bergstra and Y. Bengio. "Random Search for Hyper-Parameter Optimization". *J. Mach. Learn. Res.* 13 (2012), pages 281–305.

[15] T. Bilot, N. E. Madhoun, K. A. Agha, and A. Zouaoui. "Graph Neural Networks for Intrusion Detection: A Survey". *IEEE Access* 11 (2023), pages 49114–49139.

[16] J. Błasiok, P. Gopalan, L. Hu, and P. Nakkiran. "When Does Optimizing a Proper Loss Yield Calibration?" *Thirty-seventh Conference on Neural Information Processing Systems*. 2023.

[17] S. Bordt and U. von Luxburg. "Statistics without Interpretation: A Sober Look at Explainable Machine Learning". *ArXiv* abs/2402.02870 (2024).

[18] S. Brody, U. Alon, and E. Yahav. "How Attentive are Graph Attention Networks?" *International Conference on Learning Representations*. 2022.

[19] Y. Burda, R. Grosse, and R. Salakhutdinov. "Importance weighted autoencoders". *Proceedings of the International Conference onLearning Representations (ICLR)*. Volume 4. 2016.

[20] C. Cangea, P. Velickovic, N. Jovanovic, T. Kipf, and P. Lio'. "Towards Sparse Hierarchical Graph Classifiers". *ArXiv* abs/1811.01287 (2018).

[21] L. Chen, F. Du, Y. Hu, F. Wang, and Z. Wang. "SwinRDM: Integrate SwinRNN with Diffusion Model towards High-Resolution and High-Quality Weather Forecasting". *AAAI Conference on Artificial Intelligence*. 2023.

[22] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li. "Simple and Deep Graph Convolutional Networks". *International Conference on Machine Learning*. 2020.

[23] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li. "Simple and Deep Graph Convolutional Networks". *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. Volume 119. 2020.

[24] G. Corso, L. Cavalleri, D. Beaini, P. Liò, and P. Veličković. "Principal neighbourhood aggregation for graph nets". *Advances in Neural Information Processing Systems (NeurIPS)*. Volume 33. 2020.

[25] H. Dai, B. Dai, and L. Song. "Discriminative Embeddings of Latent Variable Models for Structured Data". *International Conference on Machine Learning*. 2016.

[26] A. K. Debnath, R. L. L. de Compadre, G. Debnath, A. J. Shusterman, and C. Hansch. "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. Correlation with molecular orbital energies and hydrophobicity." *Journal of medicinal chemistry* 34 2 (1991).

[27] A. K. Debnath, R. L. L. de Compadre, G. Debnath, A. J. Shusterman, and C. Hansch. "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. Correlation with molecular orbital energies and hydrophobicity." *Journal of medicinal chemistry* 34 2 (1991).

[28] A. Derrow-Pinion, J. She, D. Wong, O. Lange, T. Hester, L. Perez, M. Nunkesser, S. Lee, X. Guo, B. Wiltshire, et al. "ETA Prediction with Graph Neural Networks in Google Maps". *arXiv preprint arXiv:2108.11482* (2021).

[29] Y. Deshpande, S. Sen, A. Montanari, and E. Mossel. "Contextual Stochastic Block Models". *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. 2018.

[30] P. DEY, S. Merugu, and S. R. Kaveri. "Conformal Prediction Sets for Ordinal Classification". *Thirty-seventh Conference on Neural Information Processing Systems*. 2023.

[31] F. Di Giovanni, L. Giusti, F. Barbero, G. Luise, P. Lio, and M. M. Bronstein. "On over-squashing in message passing neural networks: The impact of width, depth, and topology". *International Conference on Machine Learning*. PMLR. 2023, pages 7865–7885.

[32] G. Dong, M. Tang, Z. Wang, J. Gao, S. Guo, L. Cai, R. Gutierrez, B. Campbell, L. E. Barnes, and M. Boukhechba. "Graph Neural Networks in IoT: A Survey". *ACM Transactions on Sensor Networks* 19 (2022), pages 1–50.

[33] F. Doshi-Velez and B. Kim. "Towards A Rigorous Science of Interpretable Machine Learning". *arXiv: Machine Learning* (2017).

[34] W. Du, Y. Du, L. Wang, D. Feng, G. Wang, S. Ji, C. P. Gomes, and Z. Ma. "A new perspective on building efficient and expressive 3D equivariant graph neural networks". *ArXiv* abs/2304.04757 (2023).

[35] D. Dua and C. Graff. "Adult dataset in UCI Machine Learning Repository". 2017. URL: https://archive.ics.uci.edu/ml/datasets/adult.

[36] D. Dua and C. Graff. "German dataset in UCI Machine Learning Repository". 2017. URL: https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data).

[37] I. Duta, G. Cassara, F. Silvestri, and P. Liò. "Sheaf Hypergraph Networks". *ArXiv* abs/2309.17116 (2023).

[38] T. Eimer, M. T. Lindauer, and R. Raileanu. "Hyperparameters in Reinforcement Learning and How To Tune Them". *International Conference on Machine Learning*. 2023.

[39] M. A. A. Elaziz, A. Dahou, L. M. Abualigah, L. Yu, M. Alshinwan, A. M. Khasawneh, and S. Lu. "Advanced metaheuristic optimization techniques in applications of deep neural networks: a review". *Neural Computing and Applications* 33 (2021), pages 14079–14099.

[40] M. Fey and J. E. Lenssen. "Fast Graph Representation Learning with PyTorch Geometric". *ICLR Workshop on Representation Learning on Graphs and Manifolds*. 2019.

[41] K. Fountoulakis, A. Levi, S. Yang, A. Baranwal, and A. Jagannath. "Graph Attention Retrospective". *arXiv preprint arXiv:2202.13060* (2022).

[42] C. Gallicchio and A. Micheli. "Fast and deep graph neural networks". *Proceedings of the AAAI Conference on Artificial Intelligence*. Volume 34. 2020.

[43] C. Gao, X. Wang, X. He, and Y. Li. "Graph Neural Networks for Recommender System". *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining* (2022).

[44] H. Gao and S. Ji. "Graph U-Nets". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44 (2019), pages 4948–4960.

[45] S. Garrido, S. Borysov, J. Rich, and F. Pereira. "Estimating causal effects with the neural autoregressive density estimator". *Journal of Causal Inference* 9 (2021).

[46] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. "Neural Message Passing for Quantum Chemistry". *International Conference on Machine Learning*. 2017.

[47] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. "Neural message passing for quantum chemistry". *Proceedings of the International Conference on Machine Learning (ICML)*. Volume 34. PMLR. 2017.

[48] C. Glymour, K. Zhang, and P. Spirtes. "Review of causal discovery methods based on graphical models". *Frontiers in genetics* 10 (2019), page 524.

[49] C. Gong, Y. Cheng, X. Li, C. Shan, S. Luo, and C. Shi. "Towards Learning from Graphs with Heterophily: Progress and Future". *ArXiv* abs/2401.09769 (2024).

[50] A. Graves, A.-r. Mohamed, and G. E. Hinton. "Speech recognition with deep recurrent neural networks". *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (2013), pages 6645–6649.

[51] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. "A kernel two-sample test". *The Journal of Machine Learning Research (JMLR)* 13 (2012).

[52] F. Gu, H. Chang, W. Zhu, S. Sojoudi, and L. El Ghaoui. "Implicit Graph Neural Networks". *Advances in Neural Information Processing Systems*. Volume 33. 2020.

[53] W. L. Hamilton, Z. Ying, and J. Leskovec. "Inductive Representation Learning on Large Graphs". *Neural Information Processing Systems*. 2017.

[54] S. Hochreiter and J. Schmidhuber. "Long Short-Term Memory". *Neural Computation* 9 (1997), pages 1735–1780.

[55] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec. "Open Graph Benchmark: Datasets for Machine Learning on Graphs". *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. 2020.

[56] K. Huang, Y. Jin, E. Candes, and J. Leskovec. "Uncertainty Quantification over Graph with Conformalized Graph Neural Networks". *Thirty-seventh Conference on Neural Information Processing Systems*. 2023.

[57] Q. Huang, M. Yamada, Y. Tian, D. Singh, D. Yin, and Y. Chang. "GraphLIME: Local Interpretable Model Explanations for Graph Neural Networks". *IEEE Transactions on Knowledge and Data Engineering* 35 (2020), pages 6968–6972.

[58] S. Huang, F. Poursafaei, J. Danovitch, M. Fey, W. Hu, E. Rossi, J. Leskovec, M. M. Bronstein, G. Rabusseau, and R. Rabbany. "Temporal Graph Benchmark for Machine Learning on Temporal Graphs". *ArXiv* abs/2307.01026 (2023).

[59] A. Javaloy, P. Sánchez-Martín, A. Levi, and I. Valera. "Learnable Graph Convolutional Attention Networks". *International Conference on Learning Representations (ICLR)*. 2023.

[60] A. Javaloy, P. Sánchez-Martín, and I. Valera. "Causal normalizing flows: from theory to practice". *Thirty-seventh Conference on Neural Information Processing Systems*. 2023.

[61] W. Jiang and J. Luo. "Graph Neural Network for Traffic Forecasting: A Survey". *ArXiv* abs/2101.11174 (2021).

[62] M. Jin, H. Y. Koh, Q. Wen, D. Zambon, C. Alippi, G. I. Webb, I. King, and S. Pan. "A Survey on Graph Neural Networks for Time Series: Forecasting, Classification, Imputation, and Anomaly Detection". *ArXiv* abs/2307.03759 (2023).

[63] A.-H. Karimi, J. von Kügelgen, B. Schölkopf, and I. Valera. "Algorithmic recourse under imperfect causal knowledge: a probabilistic approach". *Advances in Neural Information Processing Systems*. Volume 33. 2020, pages 265–277.

[64] I. Khemakhem, R. Monti, R. Leech, and A. Hyvarinen. "Causal autoregressive flows". *Proceedings of International Conference on Artificial Intelligence and Statistics (AISTATS)*. Volume 24. PMLR. 2021.

[65] H. Kim, S. Shin, J. Jang, K. Song, W. Joo, W. Kang, and I.-C. Moon. "Counterfactual fairness with disentangled causal effect variational autoencoder". *Proceedings of the AAAI Conference on Artificial Intelligence*. Volume 35. 2021.

[66] D. P. Kingma, T. Salimans, and M. Welling. "Improved Variational Inference with Inverse Autoregressive Flow". *ArXiv* abs/1606.04934 (2016).

[67] D. P. Kingma and M. Welling. "Auto-encoding variational bayes". *Proceedings of the International Conference on Learning Representations (ICLR)*. Volume 2. 2014.

[68] T. N. Kipf and M. Welling. "Semi-Supervised Classification with Graph Convolutional Networks". *International Conference on Learning Representations*. 2017.

[69] T. N. Kipf and M. Welling. "Variational graph auto-encoders". *arXiv preprint arXiv:1611.07308* (2016).

[70] T. Kipf and M. Welling. "Variational Graph Auto-Encoders". *ArXiv* abs/1611.07308 (2016).

[71] M. Kirchhof, B. Mucsányi, S. J. Oh, and E. Kasneci. "URL: A Representation Learning Benchmark for Transferable Uncertainty Estimates". *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. 2023.

[72] D. Klepl, M. Wu, and F. He. "Graph Neural Network-Based EEG Classification: A Survey". *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 32 (2023), pages 493–503.

[73] B. Knyazev, G. W. Taylor, and M. R. Amer. "Understanding Attention and Generalization in Graph Neural Networks". *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. 2019.

[74] B. Knyazev, G. W. Taylor, and M. R. Amer. "Understanding Attention and Generalization in Graph Neural Networks". *Neural Information Processing Systems*. 2019.

[75] M. Kocaoglu, C. Snyder, A. G. Dimakis, and S. Vishwanath. "CausalGAN: Learning Causal Implicit Generative Models with Adversarial Training". *Proceedings of the International Conference on Learning Representations (ICLR)*. Volume 6. 2018.

[76] B. Koyuncu, P. Sánchez-Martín, I. Peis, P. M. Olmos, and I. Valera. "Variational Mixture of HyperGenerators for Learning Distributions Over Functions". *International Conference on Machine Learning*. 2023.

[77] M. J. Kusner, J. Loftus, C. Russell, and R. Silva. "Counterfactual fairness". *Advances in Neural Information Processing Systems (NeurIPS)*. Volume 30. 2017.

[78] M. J. Kusner, Y. Sun, K. Sridharan, and K. Q. Weinberger. "Private causal inference". *Proceedings of the Conference on Artificial Intelligence and Statistics (AISTATS)*. Volume 19. PMLR. 2016.

[79] R. R. Lam, A. Sanchez-Gonzalez, M. Willson, P. Wirnsberger, M. Fortunato, A. Pritzel, S. V. Ravuri, T. Ewalds, F. Alet, Z. Eaton-Rosen, W. Hu, A. Merose, S. Hoyer, G. Holland, J. Stott, O. Vinyals, S. Mohamed, and P. W. Battaglia. "GraphCast: Learning skillful medium-range global weather forecasting". *ArXiv* abs/2212.12794 (2022).

[80] R. Lam, A. Sanchez-Gonzalez, M. Willson, P. Wirnsberger, M. Fortunato, F. Alet, S. Ravuri, T. Ewalds, Z. Eaton-Rosen, W. Hu, A. Merose, S. Hoyer, G. Holland, O. Vinyals, J. Stott, A. Pritzel, S. Mohamed, and P. W. Battaglia. "Learning skillful medium-range global weather forecasting". *Science* 382 (2023), pages 1416–1421.

[81] Y. LeCun, Y. Bengio, and G. E. Hinton. "Deep Learning". *Nature* 521 (2015).

[82] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel. "Handwritten Digit Recognition with a Back-Propagation Network". *Neural Information Processing Systems*. 1989.

[83] J. Lee, I. Lee, and J. Kang. "Self-Attention Graph Pooling". *ArXiv* abs/1904.08082 (2019).

[84] A. Leman. "THE REDUCTION OF A GRAPH TO CANONICAL FORM AND THE ALGEBRA WHICH APPEARS THEREIN". 2018.

[85] A. Levordashka and S. Utz. "Ambient awareness: From random noise to digital closeness in online social networks". *Computers in Human Behavior* (2016).

[86] H. Li, J. Song, L. Gao, X. Zhu, and H. T. Shen. "Prototype-based Aleatoric Uncertainty Quantification for Cross-modal Retrieval". *Thirty-seventh Conference on Neural Information Processing Systems*. 2023.

[87] L. Li, K. G. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. "Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization". *J. Mach. Learn. Res.* 18 (2016), 185:1–185:52.

[88] M. M. Li, K. Huang, and M. Zitnik. "Graph representation learning in biomedicine and healthcare". *Nature Biomedical Engineering* 6 (2022), pages 1353–1369.

[89] Q. Li, Z. Han, and X.-M. Wu. "Deeper insights into graph convolutional networks for semi-supervised learning". *Proceedings of the AAAI Conference on Artificial Intelligence*. Volume 32. 2018.

[90] X. Li, L. Sun, M. Ling, and Y. Peng. "A survey of graph neural network based recommendation in social networks". *Neurocomputing* 549 (2023), page 126441.

[91] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro. "Image Inpainting for Irregular Holes Using Partial Convolutions". *European Conference on Computer Vision*. 2018.

[92] N. Liu, Q. Feng, and X. Hu. "Interpretability in Graph Neural Networks". *Graph Neural Networks: Foundations, Frontiers, and Applications* (2022).

[93] Y. Liu, X. Ao, Z. Qin, J. Chi, J. Feng, H. Yang, and Q. He. "Pick and Choose: A GNN-based Imbalanced Learning Approach for Fraud Detection". *Proceedings of the Web Conference 2021* (2021).

[94] C. Louizos, U. Shalit, J. M. Mooij, D. Sontag, R. Zemel, and M. Welling. "Causal effect inference with deep latent-variable models". *Advances in Neural Information Processing Systems (NeurIPS)*. Volume 30. 2017.

[95] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, and X. Zhang. "Parameterized explainer for graph neural network". *Advances in neural information processing systems* 33 (2020), pages 19620–19631.

[96] P. S. Martin, T. Besold, and P. Kumari. "FRUNI and FTREE synthetic knowledge graphs for evaluating explainability". *XAI in Action: Past, Present, and Future Applications*. 2023.

[97] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. "Human-level control through deep reinforcement learning". *Nature* 518 (2015), pages 529–533.

[98] R. Moraffah, B. Moraffah, M. Karami, A. Raglin, and H. Liu. "CAN: A causal adversarial network for learning observational and interventional distributions". *arXiv preprint arXiv:2008.11376* (2020).

[99] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe. "Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks". *AAAI Conference on Artificial Intelligence*. 2018.

[100] E. Nehme, O. Yair, and T. Michaeli. "Uncertainty Quantification via Neural Posterior Principal Components". *Thirty-seventh Conference on Neural Information Processing Systems*. 2023.

[101] M. Nie, D. Chen, and D. Wang. "Reinforcement learning on graph: A survey". *ArXiv* abs/2204.06127 (2022).

[102] S. Nowozin. "Debiasing evidence approximations: On importance-weighted autoencoders and jackknife variational inference". *Proceedings of the International Conference on Learning Representations (ICML)*. Volume 35. PMLR. 2018.

[103] D. Numeroso and D. Bacciu. "MEG: Generating Molecular Counterfactual Explanations for Deep Graph Networks". *arXiv preprint arXiv:2104.08060* (2021).

[104] G. Papamakarios, I. Murray, and T. Pavlakou. "Masked Autoregressive Flow for Density Estimation". *ArXiv* abs/1705.07057 (2017).

[105] G. Papamakarios, E. T. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan. "Normalizing Flows for Probabilistic Modeling and Inference". *J. Mach. Learn. Res.* 22 (2019), 57:1–57:64.

[106] A. Parafita and J. Vitria. "Explaining visual models by causal attribution". *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. IEEE. 2019, pages 4167–4175.

[107] A. Parafita and J. Vitria. "Explaining visual models by causal attribution". *International Conference on Computer Vision Workshop (ICCVW)*. 2019.

[108] A. Parafita and J. Vitria. "Causal inference with deep causal graphs". *arXiv preprint arXiv:2006.08380* (2020).

[109] N. Pawlowski, D. Coelho de Castro, and B. Glocker. "Deep structural causal models for tractable counterfactual inference". *Advances in Neural Information Processing Systems (NeurIPS)*. Volume 33. 2020.

[110] J. Pearl. "Causal inference in statistics: An overview". *Statistics surveys* 3 (2009).

[111] J. Pearl. "Causality". *Cambridge university press*, 2009.

[112] B. Perozzi, R. Al-Rfou, and S. S. Skiena. "DeepWalk: online learning of social representations". *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (2014).

[113] J. Peters, D. Janzing, and B. Schölkopf. "Elements of causal inference: Foundations and learning algorithms". *The MIT Press*, 2017.

[114] H. T. Phan, N. T. Nguyen, and D. Hwang. "Fake news detection: A survey of graph neural network methods". *Applied Soft Computing* (2023).

[115] T. Rainforth, A. Kosiorek, T. A. Le, C. Maddison, M. Igl, F. Wood, and Y. W. Teh. "Tighter variational bounds are not necessarily better". *Proceedings of the International Conference on Machine Learning (ICML)*. Volume 35. PMLR. 2018.

[116] V. Rakesh, R. Guo, R. Moraffah, N. Agarwal, and H. Liu. "Linked causal variational autoencoder for inferring paired spillover effects". *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*. ACM. 2018.

[117] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. "Hierarchical Text-Conditional Image Generation with CLIP Latents". *ArXiv* abs/2204.06125 (2022).

[118] Y. Romano, M. Sesia, and E. J. Candès. "Classification with Valid and Adaptive Coverage". *arXiv: Methodology* (2020).

[119] Y. Rong, W. Huang, T. Xu, and J. Huang. "DropEdge: Towards Deep Graph Convolutional Networks on Node Classification". *International Conference on Learning Representations*. 2019.

[120] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "Learning representations by back-propagating errors". *Nature* 323 (1986), pages 533–536.

[121] T. K. Rusch, M. M. Bronstein, and S. Mishra. "A Survey on Oversmoothing in Graph Neural Networks". *ArXiv* abs/2303.10993 (2023).

[122] T. K. Rusch, B. P. Chamberlain, M. W. Mahoney, M. M. Bronstein, and S. Mishra. "Gradient Gating for Deep Multi-Rate Learning on Graphs". *ArXiv* abs/2210.00513 (2022).

[123] K. Sachs, O. Perez, D. Pe'er, D. A. Lauffenburger, and G. P. Nolan. "Causal protein-signaling networks derived from multiparameter single-cell data". *Science* 308.5721 (2005), pages 523–529.

[124] P. Sanchez-Martin, K. A. Khan, and I. Valera. "Improving the interpretability of GNN predictions through conformal-based graph sparsification". *ArXiv* abs/2404.12356 (2024).

[125] P. Sánchez-Martín, M. Rateike, and I. Valera. "VACA: Designing Variational Graph Autoencoders for Causal Queries". *AAAI Conference on Artificial Intelligence*. 2022.

[126] P. Sanchez-Martin, S. Utz, and I. Valera. "Exploring the Boundaries of Ambient Awareness in Twitter". 2024. arXiv: 2403.17776 [cs.SI].

[127] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. "The graph neural network model". *IEEE transactions on neural networks* 20 (2008).

[128] M. S. Schlichtkrull, N. D. Cao, and I. Titov. "Interpreting Graph Neural Networks for {NLP} With Differentiable Edge Masking". *International Conference on Learning Representations*. 2021.

[129] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. "Proximal Policy Optimization Algorithms". *ArXiv* abs/1707.06347 (2017).

[130] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. "Proximal policy optimization algorithms". *arXiv preprint arXiv:1707.06347* (2017).

[131] P. Schwab, L. Linhardt, and W. Karlen. "Perfect match: A simple method for learning representations for counterfactual inference with neural networks". *arXiv preprint arXiv:1810.00656* (2018).

[132] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra. "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization". *International Journal of Computer Vision* 128 (2016), pages 336–359.

[133] X. Shen, F. Liu, H. Dong, Q. Lian, Z. Chen, and T. Zhang. "Disentangled generative causal representation learning". *arXiv preprint arXiv:2010.02637* (2020).

[134] A. Shojaie and E. B. Fox. "Granger Causality: A Review and Recent Advances". *Annual review of statistics and its application* 9 1 (2021).

[135] J. Snoek, H. Larochelle, and R. P. Adams. "Practical Bayesian Optimization of Machine Learning Algorithms". *Neural Information Processing Systems*. 2012.

[136] Q. Sun, J. Li, H. Peng, J. Wu, Y. Ning, P. S. Yu, and L. He. "SUGAR: Subgraph Neural Network with Reinforcement Pooling and Self-Supervised Mutual Information Mechanism". *Proceedings of the Web Conference 2021*. WWW '21. Ljubljana, Slovenia: *Association for Computing Machinery*, 2021, pages 2081–2091.

[137] Z. Sun, D. Song, and A. Hero. "Minimum-Risk Recalibration of Classifiers". *Thirty-seventh Conference on Neural Information Processing Systems*. 2023.

[138] R. S. Sutton and A. G. Barto. "Reinforcement Learning: An Introduction". *IEEE Transactions on Neural Networks* 16 (2005), pages 285–286.

[139] R. S. Sutton, D. A. McAllester, S. Singh, and Y. Mansour. "Policy Gradient Methods for Reinforcement Learning with Function Approximation". *Neural Information Processing Systems (NeurIPS)*. 1999.

[140] C. Thompson. "Brave new world of digital intimacy". *The New York Times* (2008).

[141] J. Topping, F. D. Giovanni, B. P. Chamberlain, X. Dong, and M. M. Bronstein. "Understanding over-squashing and bottlenecks on graphs via curvature". *ArXiv* abs/2111.14522 (2021).

[142] G. Tucker, D. Lawson, S. Gu, and C. J. Maddison. "Doubly reparameterized gradient estimators for Monte Carlo objectives". *International Conference on Learning Representations*. 2018.

[143] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. "Graph Attention Networks". *International Conference on Learning Representations*. 2018.

[144] M. J. Vowels, N. C. Camgoz, and R. Bowden. "D'ya like DAGs? A Survey on Structure Learning and Causal Discovery". *arXiv preprint arXiv:2103.02582* (2021).

[145] M. J. Vowels, N. C. Camgoz, and R. Bowden. "Targeted VAE: Structured inference and targeted learning for causal parameter estimation". *arXiv preprint arXiv:2009.13472* (2020).

[146] Q. Wang, K. Huang, P. Chandak, M. Zitnik, and N. Gehlenborg. "Extending the Nested Model for User-Centric XAI: A Design Study on GNN-based Drug Repurposing". *IEEE Transactions on Visualization and Computer Graphics* 29 (2022), pages 1266–1276.

[147] F. R. Warburg, M. Miani, S. Brack, and S. Hauberg. "Bayesian Metric Learning for Uncertainty Quantification in Image Retrieval". *Thirty-seventh Conference on Neural Information Processing Systems*. 2023.

[148] D. Watson, J. O'Hara, N. Tax, R. Mudd, and I. Guy. "Explaining Predictive Uncertainty with Information Theoretic Shapley Values". *Thirty-seventh Conference on Neural Information Processing Systems*. 2023.

[149] B. Weisfeiler and A. Leman. "The reduction of a graph to canonical form and the algebra which appears therein". *Nauchno-Technicheskaya Informatsia* 2.9 (1968).

[150] R. Wickman, X. Zhang, and W. Li. "SparRL: Graph Sparsification via Deep Reinforcement Learning". *arXiv preprint arXiv:2112.01565* (2021).

[151] F. Wu, Q. Zhang, D. R. Radev, J. Cui, W. Zhang, H. Xing, N. Zhang, and H.-z. Chen. "Molformer: Motif-Based Transformer on 3D Heterogeneous Molecular Graphs". *AAAI Conference on Artificial Intelligence*. 2021.

[152] S. Wu, F. Sun, W. Zhang, X. Xie, and B. Cui. "Graph neural networks in recommender systems: a survey". *ACM Computing Surveys (CSUR)* (2020).

[153] X. Wu, A. Ajorlou, Z. Wu, and A. Jadbabaie. "Demystifying Oversmoothing in Attention-Based Graph Neural Networks". *ArXiv* abs/2305.16102 (2023).

[154] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip. "A Comprehensive Survey on Graph Neural Networks". *IEEE Transactions on Neural Networks and Learning Systems* (2020).

[155] K. Xia, K.-Z. Lee, Y. Bengio, and E. Bareinboim. "The Causal-Neural Connection: Expressiveness, Learnability, and Inference". *arXiv preprint arXiv:2107.00793* (2021).

[156] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. "How Powerful are Graph Neural Networks?" *ArXiv* abs/1810.00826 (2018).

[157] P. Xu, L. Zhang, X. Liu, J. Sun, Y. Zhao, H. Yang, and B. Yu. "Do Not Train It: A Linear Neural Architecture Search of Graph Neural Networks". *ArXiv* abs/2305.14065 (2023).

[158] L. Yang and A. Shami. "On Hyperparameter Optimization of Machine Learning Algorithms: Theory and Practice". *ArXiv* abs/2007.15745 (2020).

[159] M. Yang, F. Liu, Z. Chen, X. Shen, J. Hao, and J. Wang. "CausalVAE: Disentangled representation learning via neural structural causal models". *arXiv preprint arXiv:2004.08697* (2020).

[160] L. Yao, Z. Chu, S. Li, Y. Li, J. Gao, and A. Zhang. "A Survey on Causal Inference". *ACM Transactions on Knowledge Discovery from Data (TKDD)* 15 (2020), pages 1–46.

[161] R. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec. "GNNExplainer: Generating Explanations for Graph Neural Networks". *Advances in neural information processing systems* 32 (2019).

[162] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec. "Hierarchical Graph Representation Learning with Differentiable Pooling". *Neural Information Processing Systems*. 2018.

[163] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec. "GNNexplainer: Generating explanations for graph neural networks". *Advances in neural information processing systems* 32 (2019).

[164] B. Yu, H. Yin, and Z. Zhu. "Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting". *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. Volume 27. 2018.

[165] T. Yu, H. Yin, and Z. Zhu. "Spatio-temporal Graph Convolutional Neural Network: A Deep Learning Framework for Traffic Forecasting". *ArXiv* abs/1709.04875 (2017).

[166] Y. Yu, J. Chen, T. Gao, and M. Yu. "DAG-GNN: DAG structure learning with graph neural networks". *Proceedings of the International Conference on Machine Learning (ICML)*. Volume 36. PMLR. 2019.

[167] Z. Yu and H. Gao. "MotifExplainer: a Motif-based Graph Neural Network Explainer". *arXiv preprint arXiv:2202.00519* (2022).

[168] H. Yuan, J. Tang, X. Hu, and S. Ji. "Xgnn: Towards model-level explanations of graph neural networks". *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020, pages 430–438.

[169] H. Yuan, H. Yu, S. Gui, and S. Ji. "Explainability in Graph Neural Networks: A Taxonomic Survey". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45 (2020), pages 5782–5799.

[170] H. Yuan, H. Yu, J. Wang, K. Li, and S. Ji. "On Explainability of Graph Neural Networks via Subgraph Explorations". *Proceedings of the 38th International Conference on Machine Learning (ICML)*. 2021, pages 12241–12252.

[171] S. H. Zargarbashi, S. Antonelli, and A. Bojchevski. "Conformal Prediction Sets for Graph Neural Networks". *International Conference on Machine Learning*. 2023.

[172] M. Zečević, D. S. Dhami, P. Veličković, and K. Kersting. "Relating Graph Neural Networks to Structural Causal Models". *arXiv preprint arXiv:2109.04173* (2021).

[173] C. Zhang, K. Zhang, and Y. Li. "A Causal View on Robustness of Neural Networks". *Advances in Neural Information Processing Systems* 33 (2020).

[174] M. Zhang, S. Jiang, Z. Cui, R. Garnett, and Y. Chen. "D-VAE: A variational autoencoder for directed acyclic graphs". *Advances in Neural Information Processing Systems (NeurIPS)*. Volume 32. 2019.

[175] S. Zhang, A. Sohrabizadeh, C. Wan, Z. Huang, Z. Hu, Y. Wang, Y. Lin, J. Cong, and Y. Sun. "A Survey on Graph Neural Network Acceleration: Algorithms, Systems, and Customized Hardware". *ArXiv* abs/2306.14052 (2023).

[176] M. Zheng and S. Kleinberg. "Using domain knowledge to overcome latent variables in causal inference from time series". *Proceedings of the Machine Learning for Healthcare Conference (MLHC)*. PMLR. 2019.

[177] X. Zheng, Y. Liu, S. Pan, M. Zhang, D. Jin, and P. S. Yu. "Graph Neural Networks for Graphs with Heterophily: A Survey". *ArXiv* abs/2202.07082 (2022).

[178] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, and M. Sun. "Graph Neural Networks: A Review of Methods and Applications". *ArXiv* abs/1812.08434 (2018).

[179] K. Zhou, Y. Dong, K. Wang, W. S. Lee, B. Hooi, H. Xu, and J. Feng. "Understanding and Resolving Performance Degradation in Deep Graph Convolutional Networks". *Proceedings of the 30th ACM International Conference on Information & Knowledge Management* (2020).