Faculty of Engineering
Department of Electronics and Informatics

Faculty of Science and Bio-Engineering Sciences
Department of Computer Science
Artificial Intelligence Laboratory

Enhancing representation learning with Graph Neural Networks

by understanding their benefits and limitations

Dissertation submitted in fulfillment of the requirements for the degree of Doctor of Engineering Sciences

Diana Sousa Gomes

Brussels, October 2025

Promotors: Prof. Dr. Ann Nowé (Vrije Universiteit Brussel)
Prof. Dr. Peter Vrancx (Vrije Universiteit Brussel)

Members of the Jury

Prof. Dr. Ir. Kris Steenhaut	Vrije Universiteit Brussel, BE (President)
Prof. Dr. Ir. Wendy Meulebroeck	Vrije Universiteit Brussel, BE (Vice-president)
Prof. Dr. Ir. Lynn Houthuys	Vrije Universiteit Brussel, BE (Secretary)
Prof. Dr. Ann Nowé	Vrije Universiteit Brussel, BE (Promotor)
Prof. Dr. Peter Vrancx	Vrije Universiteit Brussel, BE (Co-Promotor) IMEC, BE
Prof. Dr. Ir. Nikolaos Deligiannis	Vrije Universiteit Brussel, BE
Prof. Dr. Ir. Thomas Demeester	Universiteit Gent, BE
Prof. Dr. Ir. Kevin Mets	Universiteit Antwerpen, BE

Summary

The evolution of representation learning depends not just on building more expressive models, but on aligning their design with the often messy, irregular realities of real-world data. This dissertation investigates the representational capacity of Graph Neural Networks (GNNs) through a critical and empirically grounded lens, challenging the notion that deeper or more complex models are inherently better. Instead, it advocates for a task-aware, data-sensitive approach to GNN design—one that prioritizes clarity over complexity, and rigor over trend.

Positioned at the intersection of geometric learning and deep learning, this work makes the case for rethinking how GNN architectures are developed and evaluated. It begins with a thorough literature review that critically examines the theoretical foundations of graph representation learning—particularly the implications of the curse of dimensionality, the role of geometric inductive biases, and the entanglement of learning challenges, topology and features. This theoretical analysis sets the stage for identifying what is essential for GNNs to learn effectively and how these principles should inform model design. The dissertation then transitions to a careful empirical diagnosis of the challenges posed by graphs: from their non-Euclidean structure to the wide variability of their feature content. Drawing on synthetic and real-world experiments, it demonstrates that the relative importance of structural and feature signals varies substantially across tasks—often in ways that contradict prevailing architectural assumptions. These insights expose the absence of a one-size-fits-all solution and advocate for principled, task-aware approaches to GNN development.

As such, a key contribution of this work is a principled methodology to disentangle structure and feature effects, enabling a clearer view of how GNNs actually learn. This leads to the identification of performance "sweet spots"—where graph convolutional networks excel under specific feature-structure alignments—, enabling the formulation of empirically grounded guidelines for matching GNN architectures to task and data characteristics. These insights challenge the common assumption that more complex architectures inherently lead to better performance, instead reframing the discussion around task-specific requirements.

Building on this task-sensitive perspective, we turn to one of the most debated dimensions of GNN design: depth. While depth is traditionally associated with increased capacity in neural networks, our findings suggest that this intuition does not necessarily hold for GNNs. Our analyses show that GNNs often plateau or collapse as depth increases, with layers becoming noisy or redundant. Besides creating evident computational inefficiencies, this effect can also result in behavior resembling that of a Multilayer Perceptron (MLP), in which the usage of structure is ultimately obliterated. These findings stem from the introduction of a novel evaluation protocol that reveals these failure modes in both synthetic and real-world settings. This contribution underscores the importance of embracing controlled shallowness and reinforces the broader argument of the dissertation: GNN effectiveness is highly context-dependent, and architectural decisions must be grounded in the specific characteristics of the task and data.

Ultimately, this dissertation pushes for a shift in how GNNs are understood and applied. It moves the field toward a more grounded, problem-driven mindset—where the goal is not simply to build bigger and more complex models, but to build the right ones. The resulting insights not only clarify when and why GNNs work, but also offer a practical path forward for researchers and practitioners seeking models that are better aligned with task requirements and more robust in real-world settings.

Samenvatting

De verdere ontwikkeling van representation learning vereist niet alleen expressievere modellen, maar ook het afstemmen van hun ontwerp op de vaak rommelige en onregelmatige realiteit van echte data. Dit proefschrift onderzoekt de representatieve capaciteit van Graph Neural Networks (GNNs) vanuit een kritisch en empirisch onderbouwd perspectief. Het stelt ook de gangbare opvatting dat diepere of complexere modellen per definitie beter zijn ter discussie. In plaats daarvan wordt gepleit voor een taakgerichte en data-bewuste benadering van GNN-ontwerp—een benadering die helderheid boven complexiteit stelt, en grondigheid boven trends.

Op het snijvlak van geometrisch leren en deep learning bepleit dit werk een herbezinning van de ontwikkeling en evaluatie van GNN-architecturen. Het begint met een grondige literatuurstudie waarin de theoretische fundamenten van grafrepresentaties leren kritisch worden onderzocht—met bijzondere aandacht voor de gevolgen van de curse of dimensionality, de rol van geometrische inductieve biases, en de verwevenheid van leeruitdagingen, topologie en features. Deze theoretische analyse vormt het uitgangspunt voor het vaststellen van wat essentieel is voor effectieve GNNs en hoe deze inzichten het modelontwerp zouden moeten sturen.

Vervolgens richt het proefschrift zich op een zorgvuldige empirische diagnose van de uitdagingen die grafstructuren met zich meebrengen: van hun niet-Euclidische structuur tot de grote variabiliteit in hun feature-inhoud. Op basis van synthetische en realistische experimenten wordt aangetoond dat het relatieve belang van structurele en feature-signalen sterk verschilt per taak—vaak op manieren die gangbare aannames over architectuur tegenspreken. Deze inzichten maken duidelijk dat er geen universele oplossing bestaat en benadrukken het belang van principiële, taakgerichte benaderingen in de ontwikkeling van GNNs.

Een belangrijke bijdrage van dit werk is dan ook een methodologie om structuur- en feature-effecten van elkaar te ontkoppelen, wat een helderder inzicht biedt in hoe GNNs daadwerkelijk leren. Dit leidt tot de identificatie van prestatie-‘sweet spots’—situaties waarin graf convolutionele netwerken uitblinken bij specifieke combinaties van features en structuur.Dit maakt het mogelijk om empirisch onderbouwde richtlijnen op te stellen voor het afstemmen van GNN-architecturen op taak- en data-eigenschappen. Deze inzichten gaan in tegen de gangbare veronderstelling dat complexere architecturen automatisch tot betere prestaties leiden, en herdefiniëren het debat rond de specifieke eisen van een taak.

Vanuit dit taak-sensitieve perspectief richt het proefschrift zich vervolgens op een van de meest besproken dimensies van GNN-ontwerp: diepte. Hoewel diepte in neurale netwerken traditioneel wordt geassocieerd met verhoogde capaciteit , wijzen onze bevindingen uit dat deze intuïtie niet vanzelfsprekend opgaat voor GNNs. Onze analyses tonen aan dat GNNs vaak een prestatiedrempel bereiken of zelfs instorten naarmate de diepte toeneemt, waarbij lagen steeds meer ruis en redundantie bevatten. Dit leidt niet alleen tot computationele inefficiëntie, maar kan er ook toe leiden dat het model zich gedraagt als een Multilayer Perceptron (MLP), waarbij het gebruik van structuur volledig verloren gaat. Deze bevindingen vloeien voort uit een nieuw evaluatieprotocol dat deze faalpatronen blootlegt in zowel synthetische als realistische omgevingen. Dit benadrukt het belang van gecontroleerde gebruik van diepteen versterkt de bredere these van het proefschrift: de effectiviteit van GNNs is sterk contextafhankelijk, en architecturale keuzes moeten verankerd zijn in de specifieke kenmerken van taak en data.

Dit proefschrift pleit uiteindelijk voor een paradigmaverschuiving in hoe GNNs worden begrepen en toegepast. Het veld wordt hiermee richting een meer probleemgestuurde en onderbouwde benadering geleid—waarbij het doel niet is om simpelweg grotere en complexere modellen te bouwen, maar om de juiste modellen te ontwikkelen. De resulterende inzichten verduidelijken niet alleen wanneer en waarom GNNs effectief zijn, maar bieden ook een praktische leidraad voor onderzoekers en gebruikersdie op zoek zijn naar modellen die beter aansluiten bij taakspecifieke vereisten en robuuster zijn in echte toepassingen.

Lay Summary

Graphs—structures of points linked by relationships—are everywhere: in social media, transportation systems, biology, and more. Graph Neural Networks (GNNs) are a new kind of artificial intelligence designed to learn from these complex connections. But despite their popularity, we still do not fully understand when they work best or why they sometimes fail.

This dissertation takes a closer look at how GNNs actually learn. It shows that their success depends on the delicate balance between the information in the graph's connections and the data attached to its points. We introduce new tools to separate and measure these two factors, revealing situations where simpler methods can outperform even the most advanced GNNs. The common belief that "deeper" networks are always better is also challenged, and we demonstrate that adding layers often adds noise instead of insight.

By identifying these limitations and offering practical guidelines for designing GNNs, this work helps researchers and engineers build models that are not just more powerful, but also more reliable and efficient—paving the way for smarter AI systems that can handle the complex relational problems that shape our world.

Acknowledgments

This thesis looks deeply into networks of connected points, but no network was as significant to me as the one of relentless support, share of knowledge, and love that these two pages represent, and without whom this work would not exist.

First, and foremost, I would like to thank my advisors, Peter Vrancx and Ann Nowé. Peter, with whom not only I had the privilege to learn tremendously on a daily basis, but who always pushed me to aim higher while staying true to my interests and scientific curiosity. I am a better researcher on all accounts because of you. Thank you for the countless discussions, readings, materials, guidance, and—perhaps even more determinant—never leaving me behind or without a sense of direction over four years of fast-paced changes. This thesis is also the tale of department moves, company-wide reorganizations, new roadmaps and fast reshaping of goals; but you provided me with constant professional and personal support from day one. I am very privileged and forever grateful for that. And Ann, for giving me the opportunity of learning from your experience and for your sustained guidance. You helped me navigate the world of academia and pushed me to communicate my research with higher quality and purpose. Thank you for the invaluable feedback, especially in this final year.

I would also like to thank all members of the doctoral examination committee, Kris Steenhaut, Wendy Meulebroeck, Lynn Houthuys, Nikolaos Deligiannis, Thomas Demeester and Kevin Mets, for your time and interest, the productive discussions and positive feedback, and the overall suggestions that improved the quality of this work.

Embracing the PhD journey made me take big steps, such as moving countries, and gain a whole new perspective on my career and how I want to contribute to this world. And that is in great measure due to the professional connections I established in this period, in both IMEC and VUB. My first gateway into life in Belgium was the CSA team. You welcomed me, taught me how to communicate my research in angles I had never explored, and brought people into my life who I still cherish as some of my closest friends—thank you. More recently, a new team crossed my path, AIA, where I found a completely different hub of AI expertise—one where my application-driven senses could also flourish, marking yet another important milestone of this dissertation. Thank you for welcoming and supporting me, and for the continuous knowledge exchange. At the VUB, I also established a set of very important connections from the AI Lab. In particular, Kirk, who closely helped setting up the first stages of this PhD and provided me with invaluable feedback towards steering my research; Roxana, whose work dedication and thirst for knowledge inspire me and with whom I keep learning so much; and Hicham, who is always up-to-date with the latest

AI trends and shares this knowledge so effectively. Thank you for the best reading group I have experienced—I am a more skilled researcher because of you.

I was also fortunate enough to find several friends along these years that not only brightened my (often cloudy) days, but with whom I shared several milestones, achievements, and frustrations of PhD life. Emil, Nitish, Giuseppe, thank you for the countless chats, for making me feel understood in my challenges, and for your friendship—it seems like just yesterday we all started this journey, and now I am happy to be the one shutting the door behind me. Nathan, I enjoyed every single conversation on stochasticity, soup, and even the meaning of life; but beyond that, your perseverance of the past few years was and still is an inspiration for me—thank you. Joana S., my twin of busy agendas and defense dates, you took me in from the day we met and, since then, we never let go. Thank you for all the hugs and tears, couches and concerts, and an infinite number of stories to tell.

Life has this way of making us keep the important people that cross our path, no matter how far they are, or for how long we have been close or apart. This network shapes who I am and what I am able to achieve. Jean, thank you for the many "middle brother" deep talks. Sara, thank you for bringing your lightness, good advice, and undivided attention every time we meet. Sevana, thank you for each coffee with a profound conversation. Ana, thank you for being my frequent travel buddy during the most difficult periods of expat life. Ann-Kristin, thank you for all the meetups across Europe that made me better enjoy the perks of living in Belgium. Pife, thank you for the endless personal, technical, professional, and overall relentless support over so many years. Joana C., thank you for being my "big sister" of academic life, for the daily complaining, and for my cosy bed in Lisbon. Tiago, thank you for all the care in the form of a great meal and some nice bottles of wine. Fábio, thank you for being a present, caring friend for so many years, for each ride, each beer and each pool game. Manuela, thank you for all of the adventures growing up and for being my unconditional example of resilience. Daniela, thank you for knowing me like the back of your hand and sharing all the good and bad moments of a lifetime.

Kalliopi, your friendship and constant, never-ending support is one of the reasons why this milestone could ever exist. Thank you for making me feel at home at all times, even in the long winters in which we are already imagining the summer again. I could have never asked for a better neighbor and friend. Mariana, I cannot count the number of times we went over the ups and downs of PhD life over the past few years. Thank you for always picking up the phone, for the long talks that "solve life", for all the adventures, and for the unconditional friendship.

To my family who gives up on having me around all year long so that I can pursue my goals, my most heartfelt thank you. To my Tio Albino and Tia Alice, Irene, and Pedro, thank you for always believing in me and pushing me to achieve everything I aspire to since childhood. To Salomé, who keeps being the most present and loving sister I could have ever hoped for. E aos meus pais, cujo apoio incondicional e sacrifício pessoal me trouxeram até aqui.

And finally, to Giuliano, who witnessed the struggles of every day from a front row seat and filled my days with care and support—thank you for the patience, for believing in me, and for being my partner and best friend.

List of Abbreviations

Abbreviation	**Definition**
AI	Artificial Intelligence
ANN	Artificial Neural Network
BN	Batch Normalization
CE	Cross-Entropy
CKA	Central Kernel Alignment
CNN	Convolutional Neural Network
GCN	Graph Convolutional Network
GCN+InitRes	Graph Convolutional Network with Initial Residual Connection
GNN	Graph Neural Network
LSTM	Long Short-Term Memory
ML	Machine Learning
MLP	Multi-Layer Perceptron
MPNN	Message Passing Neural Network
MSE	Mean Squared Error
NAS	Neural Architecture Search
PyG	PyTorch Geometric
RNN	Recurrent Neural Network
SBM	Stochastic Block Model
SGC	Simple Graph Convolution
SGD	Stochastic Gradient Descent
SNR	Signal-to-Noise Ratio
s.s.d.	Statistically significant difference
UAT	Universal Approximation Theorem

List of Figures

List of Tables

Contents

1 Introduction

A world of data

With the advent of the internet and the establishment of powerful machine learning models as ubiquitous in modern society, the world has become increasingly more data-driven. Data are the basis that enable machines to learn from experience. Trying to make machines learn a representation of the world depends on two crucial aspects: 1) how that world is translated into machine-tractable quantities; and 2) how to process them to extract insightful patterns.

Take the example of computer vision tasks, one of the historically successful cases of machine learning. Images consist of a matrix of values, i.e. pixels. These units enable the digitization of the continuous world we see into a discrete set of values that machines can process. Similarly, text can be discretized into sequential tokens of words and time-dependent phenomena into quantities sampled at regular time intervals (e.g. amplitudes for audio or acceleration values for movement). The relation between these data units is the basis for the extraction of relevant patterns, typically encoded into abstract representations by machine learning models and finally decoded back into human-interpretable quantities with problem-solving abilities.

Relational data Currently, most machine learning toolboxes are designed to operate in sequences or grids like the cases described above. But not all cases in which data points have a relational dependency can benefit from such structure regularity and order. Many

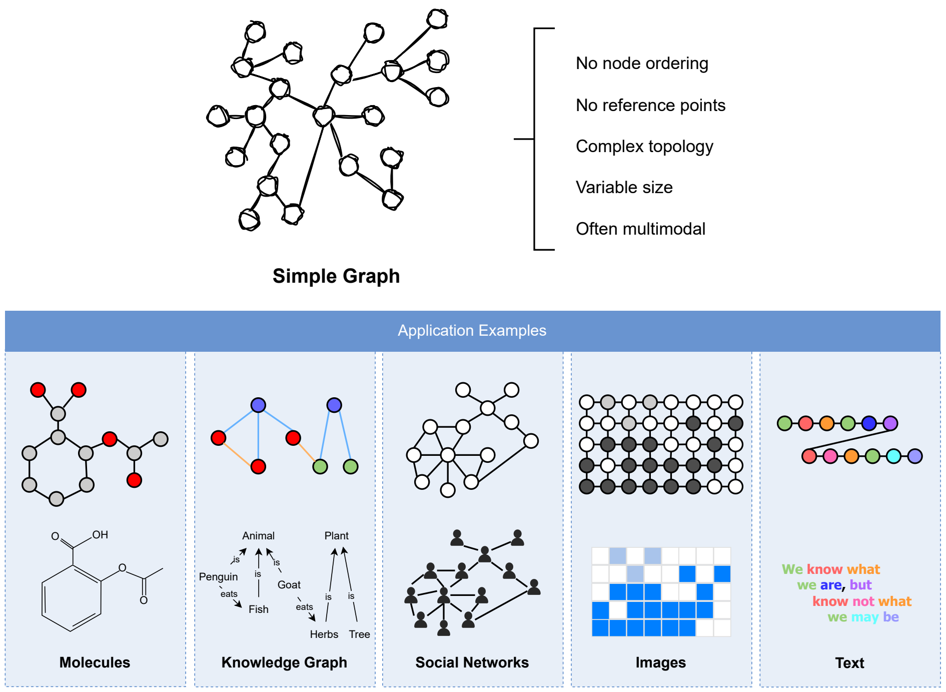


Figure 1.1: Graph structures across real-world domains and their inherent complexity. [Top] Core characteristics that make graph learning distinct and challenging. Together, these features underscore both the power of graph representations and the unique difficulties they pose for machine learning models, motivating the development of specialized methods to effectively learn from such data. [Bottom] Graph-structured data naturally arises in a wide range of real-world applications, including *molecular graphs* (where atoms are nodes and bonds are edges), *knowledge graphs* (entities and their relationships), *social networks* (individuals connected by social ties), *images* (modeled as pixel graphs), and *text* (represented as sequential graphs, co-occurrence graphs, or other structured forms). Domain-specific learning tasks can be defined over these graphs (e.g., property prediction, entity classification, link prediction).

complex domains depend on important relational information which can only be fully captured by a more abstract category of data structures: *graphs*.

In its simplest form, a *graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a set of elements $\mathcal{V}$, called nodes (or vertices), and a set of unordered pairs of nodes $\mathcal{E}$, called edges (or links). In other words, nodes are the entities that interact with each other; and those interactions are depicted by edges. Due to this very general definition, graphs are very broadly applicable, being able to unify the representation of all types of relational data, including grids, sequences, Euclidean spaces, and further special cases (Figure 1.1).

This unified view of very different rich domains creates significant opportunities for the development of learning methods operating on graphs, due to their potential in becoming relevant across all relational domains – *a new frontier of machine learning*.

Representation Learning on Graph Data

This dissertation aims to *advance representation learning on graphs* by emphasizing that its impact in both foundational research and real-world applications depends critically on the specific characteristics of graph problems and the tailored optimization of learning algorithms. We posit that relational data—embodied in graphs—offers a universal descriptive framework for complex systems, but the diversity of problems that can be defined under such a framework (such as the examples in Figure 1.1) poses severe challenges to the development of effective methods across the spectrum. Despite these challenges, we envision graph learning evolving into a general-purpose methodology for representing, reasoning about, and learning from structured data in its many forms. The question that remains is: *how do we get there?*

Our Scope *Learning on graphs*, also referred to throughout this dissertation as *graph learning*, in line with common terminology in the literature, denotes the class of machine learning methods that operate over data represented as graph structures. In this context, both terms will be used interchangeably to signify algorithms and models designed to process, analyze, and make predictions based on data encoded in the form of nodes and edges, which capture complex relationships and interactions between entities. Importantly, the focus here is on *leveraging existing graph topologies to learn patterns or perform inference, rather than generating new graph structures.* This can include, but is not limited to, tasks such as node classification, link prediction, and graph-level classification.

Moreover, within the broader scope of graph learning, this dissertation adopts the term *graph representation learning* to specifically refer to the task of learning meaningful representations, or embeddings, over graph-structured data. This encompasses learning encodings at multiple levels of granularity, including node-level embeddings that capture local neighborhood information, edge-level representations that model pairwise relationships, and whole-graph representations that summarize global structural and semantic properties. These learned representations serve as compact, informative features that can be leveraged for downstream tasks such as classification, clustering, link prediction, and regression.

As a foundational component of graph learning, graph representation learning plays a critical role in bridging the gap between raw graph data and the performance of machine learning models. Despite substantial progress, the field continues to face several open challenges. These include developing architectures that can generalize across diverse graph domains, creating highly discriminative representations at the different levels of granularity, scaling up methods, and integrating domain-specific inductive biases without compromising generality. This dissertation engages with these challenges, and proposes to bring together the theoretical and practical sides of graph representation learning as key pillars to enable intelligent systems that understand and reason appropriately over structured data.

To set the stage for the contributions of this dissertation, let us turn to a historical perspective on the developments of this promising field.

Learning on Graphs: a Historical Perspective

Graph learning is a subfield of machine learning dedicated to graph-structured data, and it has been evolving over several decades, even centuries. Graphs have been studied in mathematics since their first proposition by Leonhard Euler in 1736 to address the problem of the "Seven Bridges of Köningsberg" on whether it was possible to cross each of the seven bridges exactly once while walking through the city. By abstracting the representation of the city as a graph, where land regions were depicted as nodes and bridges were depicted as edges, Euler proved that such a walk was impossible [Euler, 1741].

The work of Euler laid the foundations for what we now know as *graph theory* and, thus, modern graph algorithms can trace back their origins to as early as the 18th century. But it was only in the late 20th century that the expansion of this field alongside computer science led to revolutionary *computational graph analysis algorithms*, achieving new significant milestones.

Traditional Graph Learning PageRank is an example of such key foundational works. This algorithm was proposed by Larry Page and Sergey Brin in 1996, and it laid the groundwork for what would become Google's search algorithm. By modelling the web as a graph, where nodes represent web pages and edges represent hyperlinks between them, they were able to approximate the importance of a web page by how many other important pages link to it and rank them accordingly [Page et al., 1999], dramatically improving search relevance. The importance of this work extends beyond the context of search engines. PageRank was one of the first large-scale, real-world applications of graph-based ranking algorithms, the idea of message propagation and spectral graph methods, influencing many algorithms for learning on graphs to this day.

This and other classical methods are often broadly categorized as *traditional graph learning*. Typically, traditional graph learning methods rely on and/or combine handcrafted features, probabilistic models (e.g., PageRank [Page et al., 1999]), random walk-based methods (e.g., DeepWalk [Perozzi et al., 2014], node2vec [Grover and Leskovec, 2016]), spectral methods (e.g., Laplacian Eigenmaps [Belkin and Niyogi, 2003]), matrix factorization methods (e.g., LINE [Tang et al., 2015], HOPE [Ou et al., 2016]), and further approaches to generate meaningful representations of graph-structured data and ultimately extract relevant patterns from them.

Towards *Automatic* Graph Representation Learning These approaches represent the foundational efforts in *graph representation learning*, encoding graph-structured data into low-dimensional vector spaces while preserving important topological and relational information of the original graph.

Despite their success for specific problems and the significant advances that they inspired, these techniques are generally known for scaling poorly—even PageRank, arguably one of the most scaled algorithms ever, required significant investment and innovations in infrastructure and distributed computing to enable massive-scale indexing and ranking—and for their limited generalization capabilities. Spectral methods, for example, use eigenvectors of the graph Laplacian to create node embeddings, preserving important local information; however, eigendecomposition is a highly costly operation, preventing these methods from handling large graphs. Random walk-based methods improved scalability while still capturing neighborhood information efficiently by local sampling; but this set of methods lacks transductive learning abilities, thus rendering them inapplicable to unseen nodes. Matrix factorization methods also scale better to large graphs than the spectral ones, but they are still memory and computation hungry and their expressiveness is theoretically reduced due to the sole preservation of linear transformations and the complete obliteration of non-linearities.

Moreover, traditional graph learning methods heavily rely on manual feature engineering, where domain-specific features are designed to capture the most relevant structural properties of graphs for a certain task. This process is typically time-consuming and requires a high amount of expertise in the application domain, manifesting as a major bottleneck to fast and efficient model development. It is in this context that *Graph Neural Networks* (GNNs) emerge, marking a paradigm shift towards automated feature extraction and representation learning through end-to-end training and, ultimately, revolutionizing the field.

Deep Learning on Graphs GNNs emerged as the go-to graph representation learning approach at the interface of graph learning and deep learning following the seminal work of Kipf and Welling [2017] on *Graph Convolutional Networks* (GCNs). However, the roadmap towards their establishment can be traced back to the early 2000s, with the works of Gori et al. [2005] and Scarselli et al. [2009] where the term was initially coined. With the rise of deep learning, particularly following the introduction of AlexNet, research into applying deep learning techniques to graph-structured data also gained attention and prominent approaches have emerged from spectral methods (e.g., Spectral CNNs [Bruna et al., 2014]). These works laid the groundwork for the disruptive innovation of GCNs — an efficient way to approximate spectral graph convolutions with spatial graph convolutions, making them computationally feasible for large-scale graphs.

Traditional spectral graph convolution relies on the eigenvectors and eigenvalues of the graph Laplacian, transforming signals into the spectral domain for filtering. However, computing the full Laplacian eigenbasis is costly, prompting Kipf and Welling [2017] to introduce a first-order Chebyshev polynomial approximation that enables localized feature propagation without explicit eigenvector computation. This reformulation aligns with a message-passing framework, where nodes aggregate information from their immediate neighbors. Avoiding eigendecomposition significantly reduces computational complexity from $O(|\mathcal{V}|^3)$ (cubic in the number of nodes) to $O(|\mathcal{E}|)$ (linear on the number of edges), making it suitable for large-scale graphs while preserving localized feature representations, which is particularly beneficial in semi-supervised learning tasks. Moreover, its adjacency renormalization *trick* enhances numerical stability, and its compatibility with standard deep learning frameworks facilitates efficient implementation with matrix multiplications.

This elegant balance between spectral theoretical rigor and spatial computational efficiency promoted the widespread adoption and application of different forms of GCNs in several different domains, such as drug design [Wong et al., 2024], biology [Jha et al., 2022] and complex physics simulations [Sanchez-Gonzalez et al., 2020]. Inspired by their success, new network architectures have been proposed in the following years aiming to improve expressivity [Veličković et al., 2018; Xu et al., 2019], decrease computational cost [Hamilton et al., 2017], handle graph oversmoothing [Chen et al., 2020; Zhao and Akoglu, 2019] and overcome harmful node-level heterophily [Yan et al., 2022; Pei et al., 2019; Bodnar et al., 2022]. These are still areas of active research, and will be revisited in the next chapters.

Achievements in Key Application Domains Graph learning has led to breakthroughs across multiple high-impact domains. Among these achievements is the work of the AlphaFold team on protein structure prediction, recognized by the 2024 Nobel Prize in Chemistry [Jumper et al., 2021]. While AlphaFold does not directly use GNNs, it models proteins as relational structures and applies geometric deep learning principles. This highlights how learning from structured data, including graphs, is reshaping scientific discovery and opening new opportunities for AI-driven biomedical research. Further remarkable innovations can also be highlighted in the following fields:

- ***Social Networks and Recommendation Systems.*** Scalable, inductive graph learning has enabled the generalization of recommendation models to unseen users [Hamilton et al., 2017; Ying et al., 2018a], powering platforms such as Facebook, Twitter, TikTok, YouTube and LinkedIn.
- ***Drug Discovery and Biomolecules Modelling.*** Representing molecules as graphs has led to significant advances in accelerating drug discovery and understanding natural phenomena by predicting molecular properties and interactions, protein folding and molecular dynamics through graph learning [Jumper et al., 2021; Ying et al., 2021; Hoogeboom et al., 2022].
- ***Fraud Detection and Financial Networks.*** By handling financial networks, user-transaction graphs and other relational entities, GNNs have improved fraud/anomaly detection for banking systems and platforms like PayPal and Alipay [Wang et al., 2019a] [Wang et al., 2024].
- ***Natural Language Processing (NLP) and Knowledge Graphs.*** Graph-based knowledge representations have led to better reasoning schemes, improving question answering, search and recommendation (Google, OpenAI) [Bordes et al., 2013; Wang et al., 2019b; Agarwal et al., 2021].
- ***Computer Vision and Image Processing.*** Graphs have been used to model object relationships and promote scene understanding, enabling scene graph generation and 3D object recognition in applications such as autonomous driving and robotics [Santoro et al., 2017; Yang et al., 2018].
- ***Traffic and Smart Cities.*** Graph-based city mapping has been enabling groundbreaking real-time traffic predictions/forecasting (e.g., Google Maps), urban planning and air quality/meteorology predictions [Yan et al., 2018; Derrow-Pinion et al., 2021; Li et al., 2023].

Building on the historical trajectory and real-world successes of graph learning, we now turn to *examine the recent evolution of GNN research.* While the field has made remarkable technical progress—introducing increasingly powerful models and expanding its application scope—the rapid pace of growth has also led to fragmentation, conceptual ambiguity, and conflicting empirical findings, as we will see in the next subsection.

This dissertation takes the view that, to unlock the full potential of GNNs, it is essential to critically reassess the foundations of recent advances, clarify common misconceptions, and develop a more principled understanding of how model design, problem formulation, and evaluation strategies interact. It is within this context that the contributions of this thesis are positioned—aiming to bring clarity, structure, and targeted improvements to the evolving landscape of graph representation learning.

GNN Research: Advances and Challenges

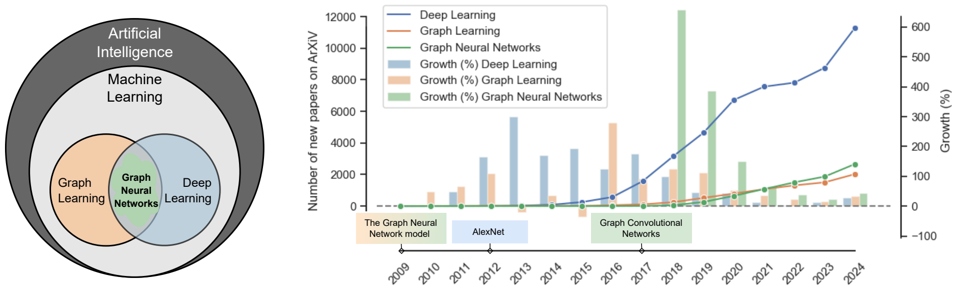


Figure 1.2: Relation between Graph Learning, Deep Learning and GNN as subfields of Machine Learning [left] and their growth trends measured in number of new ArXiV papers per year [right]. Foundational works are also chronologically shown, evidencing their impact in the growth of their field over the following years [Scarselli et al., 2009; Krizhevsky et al., 2012; Kipf and Welling, 2017]. Data were extracted from [Batalha, 2025] using the keywords and method described in Appendix A.1.

GNN research has experienced significant growth in recent years. Figure 1.2 illustrates this rapid expansion, situating it alongside landmark developments that marked this modern deep learning era. As of 2024, research in this field continues to accelerate, with approximately 2.600 GNN papers being submitted to ArXiv annually—an average of ~7 new papers per day. Recent efforts have concentrated mostly on enhancing both predictive performance and computational efficiency of GNNs for a wide range of graph problems. In parallel, self-supervised learning approaches, such as contrastive learning and diffusion-based models, are gaining traction for their potential to yield more generalizable representations and to unlock novel generative capabilities.

The data in Figure 1.2 further reveal that both deep learning and graph learning have undergone dramatic growth spurts around pivotal years, with the number of new publications more than doubling within a single year on certain occasions. This trend is particularly striking for GNNs, which experienced a surge of over 600% in newly proposed methods and applications in 2018—the year following the publication of GCNs. This remarkable inflection point marked the beginning of a sustained period of rapid expansion, underscoring the transformative impact of GCNs and the growing interest in leveraging graph-structured data across a wide range of domains.

While these years have brought significant advances in the scalability and expressivity of GNNs, the rapid pace of development has also introduced a degree of ambiguity and, at times, contradictory findings across various aspects of model behavior and design. These challenges will be revisited in the next subsections, where we argue that much of the current confusion stems from a lack of rigorous analyses and the frequent neglect of essential empirical validation steps—shortcomings often driven by the pressure to produce competitive results in a fast-paced research environment. We will explore how this dilemma shapes the current state of GNN research, its broader implications and their impact going forward, steering us towards the research question of this work.

The Pains and Gains of Rapid Growth

The revolution brought by GCNs was not a single work's effort. Posterior works on scalability, inductive capabilities and network expressivity were also critical to explore the potential of GNNs in all levels of graph learning tasks. In particular, it is possible to highlight the impact of GraphSAGE [Hamilton et al., 2017] (improve scalability by introducing a neighborhood sampling approach), GAT [Veličković et al., 2018] (implement attention mechanisms in the message aggregation framework), GIN [Xu et al., 2019] (effectively distinguish certain different graph structures, setting a new standard for expressiveness in GNNs), and SGC [Wu et al., 2019] (simplify the GCN model by removing nonlinearities and collapsing weight matrices of consecutive layers).

GNNs' Shortcomings These publications have collectively advanced the development and application of GNNs, providing foundational methodologies and insights that continue to influence current research and practice. Nonetheless, these are not yet sufficient to address all the challenges surrounding GNNs. GNNs have been found to perform poorly in several cases. The causes for this behavior are generally assigned either to oversmoothing, oversquashing, or the challenging structure of the input graph (frequently some harmful cases of heterophily). Given their high impact, these causes quickly became the focus of intensive research with consecutive new methods proposing to mitigate their damaging effect in performance.

Oversmoothing, where increasing the number of layers causes node embeddings to become indistinguishable, degrading model performance, is likely the phenomenon that has received more attention from the research community [Yan et al., 2022; Balcilar et al., 2021; Keriven, 2022; Oono and Suzuki, 2019; Rusch et al., 2023a]. Traditional metrics, like the Dirichlet energy, have been used to quantify this effect, but they have limitations and may not reliably measure if a method appropriately mitigates it in practical scenarios [Rusch et al., 2023a], leading to ambiguous or insufficient conclusions on the effectiveness of current strategies. Another issue is oversquashing, where GNNs struggle to propagate information across long-range node interactions, leading to information bottlenecks.

The causes, consequences, and mitigation strategies for oversquashing, including graph rewiring and curvature-based strategies, continue to be investigated and are frequently considered intertwined with the causes/effects of oversmoothing [Akansha, 2025; Giraldo et al., 2023]. Node-level heterophily—where connected nodes have dissimilar features—has also been pointed out as a cause for poor performance. Similarly to oversquashing, its impact has been considered on its own but also with relation to oversmoothing as "two sides of the same coin" [Yan et al., 2022; Giovanni et al., 2023; Bodnar et al., 2022].

Architectural Design Choices The rapid proliferation of GNN architectures has also led to diverse design choices, frequently lacking consensus regarding their applicability. For example, while non-linearities are considered to be critical for the expressiveness of neural networks, simplified linear graph convolution methods have been shown to maintain comparable performance to the original GCNs in a variety of node classification tasks [Wu et al., 2019], hinting at the fact that the biggest added-value of this message-passing scheme might be the aggregation itself and not the non-linear transformation of the node embeddings. Despite these results, non-linear GNNs are currently significantly predominant.

Similarly, there is controversy regarding the role of depth, residual connections, attention, and other design choices, including how they relate to each other and to the properties of the input graph. While some advanced network architectures can go deep without apparent oversmoothing [Chen et al., 2020; Yan et al., 2022; Rusch et al., 2023b], these deep GNNs have brought little to no practical gains in terms of performance. Inspired by their success in other deep learning domains, residual connections have been used to enable deeper networks, but it is unclear in which form they are needed or essential [Jaiswal et al., 2022] and whether their contribution relates to their sharpening effect, preventing vanishing gradients, or further phenomena. Moreover, concerning attention, while some studies assert that attention mechanisms in GNNs can mitigate oversmoothing [Lee et al., 2023], rigorous mathematical analysis indicates that attention-based GNNs still lose expressive power exponentially with depth, similar to traditional GCNs [Wu et al., 2023].

Overall, the causes and mitigation strategies for oversmoothing and oversquashing also remain unsettled, with conflicting evidence on whether architectural modifications, such as attention mechanisms or alternative aggregation functions, consistently improve performance. Similarly, the impact of heterophily on GNNs is disputed, as some studies suggest that modified aggregation strategies enhance learning in heterophilous graphs [Yan et al., 2022], while others find these solutions highly context-dependent [Luan et al., 2023]. These contradictions underscore the complexity of GNN research and highlight the necessity for comprehensive evaluation frameworks, diverse benchmarks, and rigorous theoretical analyses to reconcile differing findings and advance the field.

Empirical Research in (Graph) Machine Learning: where are we?

This scenario of confusion and contradiction is not unique to GNNs or graph learning; in fact, it is a persistent problem in machine learning research. Herrmann et al. [2024] call it *"the non-replicable machine learning enigma"*. The authors argue that current practices make empirical machine learning research often prone to overly optimistic and unreliable conclusions due to insufficiently operationalized experimental setups and ambiguous conceptual foundations. To improve reliability, more explicit and context-specific operationalizations are needed, especially in deep learning, where the complexity of modern models necessitates rigorous experimental analysis.

Empirical machine learning stands at the intersection between formal sciences and real-world applications and involves the systematic investigation of algorithms, techniques, and conceptual questions through simulations, experimentation, and observation, emphasizing real-world implementations rather than purely theoretical analysis [Herrmann et al., 2024]. This approach requires a distinct methodological mindset that fully accounts for the uncertainties inherent to working with real-world data and computational systems, and its contributions are critical for scientific breakthroughs.

For these reasons, Herrmann et al. [2024] analyze a significant body of literature and make a plea for (quoting) *"more insight-oriented exploratory research"*—open-ended approach that seeks to generate insights, identify patterns, and formulate hypotheses in areas where little prior knowledge exists—and *"more (actual) confirmatory research"*—structured, hypothesis-testing approach that evaluates preexisting theories through predefined study designs and statistical analyses to draw conclusive inferences.

If we take the example of GNN research, this could mean comparing different methods/network architectures under the same rigorous evaluation framework to bring insights on topics that lack consensus, such as the (mitigation of) oversmoothing or the impact of graph structure in performance degradation, following a dedicated design of experiments aiming at verifying an initial hypothesis, or creating and disseminating tools to assist with this process, whether benchmarks or validation code bases. These practices could break the cycle of misunderstandings surrounding GNNs, their benefits and limitations.

Understanding and Advancing Representation Learning with GNNs: Motivation and Research Questions

This broad context places GNNs in a nuanced and critical position. While these networks are attractive on many levels—algorithmic elegance and simplicity, computational feasibility and compatibility with general deep learning frameworks, end-to-end trainable for all types of graph learning tasks (i.e., node-level, edge-level, graph-level)—, they also frequently fail or underperform in comparison with more traditional methods. This poses pivotal questions about the boundaries of their benefits and limitations, what is critical for them to learn outstandingly, and how to turn this knowledge into practice. These questions remain insufficiently addressed to date.

Efforts towards creating unified frameworks of geometric deep learning have been made, thoroughly defining the symmetries of the data and mapping them with respect to adequate network architectures that preserve such symmetries [Bronstein et al., 2021]. However, as we will see in the next chapter, these theoretical frameworks are insufficient towards defining the universe of unique characteristics of GNNs in comparison to other neural networks (as they only focus in the geometric part of the learning problem), and can mislead researchers towards believing that conventional deep learning design principles can directly translate to graph-based architectures, provided that the base requirement of symmetry preservation is satisfied.

Confident that appropriate empirical research practices are a fundamental pillar of scientific breakthrough, this work aims to pave the way towards enhancing graph representation learning by measuring the practical benefits and limitations of current GNN methods. In particular, it proposes to answer the following questions:

What are the key requirements for the effective performance of GNNs? And which design choices can make GNNs learn better representations, more efficiently?

Having established the motivation and central research questions guiding this work, the following section outlines the core contributions of this dissertation. These contributions aim to address the identified gaps by combining theoretical insight with rigorous empirical investigation, ultimately advancing our understanding of what drives effective and efficient representation learning with GNNs. Through targeted analyses and methodical experimentation, this thesis seeks to offer both conceptual clarity and practical guidance for the development of more robust, interpretable, and high-performing GNNs.

Thesis Contributions

This thesis operates at the interface of graph representation learning, deep learning architecture design, and empirical machine learning methodology to align widely adopted approaches for learning on graphs, particularly GNNs, with a clearer understanding of the practical benefits and limitations associated with their design choices. It exposes the cases in which GNNs fail and succeed in light of their theoretical foundations, but from an empirical point of view with dedicated design of experiments aimed at (dis-)proving well-bounded hypotheses—a perspective that is frequently missing in the literature and shall influence future research directions by contributing to *confirmatory research* [Herrmann et al., 2024]. Given this context, this manuscript advances the following thesis:

Thesis

GNNs are a particular type of neural network and one should not expect them to benefit from the same design choices as deep networks with different geometric inductive biases by default. *Systematic hypothesis-testing research practices can disclose the practical benefits and limitations of popular GNN design choices*, effectively assisting researchers in the problem definition and the optimization of network architectures and learning tasks. These efforts can significantly contribute to understanding how to *fully leverage GNNs' capacity to solve complex tasks on relational data optimally while maximizing the discriminative power of the learned representations.*

The next paragraphs will systematically break down these overarching contributions into more specific components, providing a detailed examination of each aspect and framing them with respect to the chapters of the manuscript.

At the Interface between Graph and Deep Learning: a unifying (re)view Chapter 2 positions itself at the dynamic intersection between graph learning and deep learning, emphasizing the need to reconcile mathematical foundations with practical demands. Through a thorough examination of the geometric inductive biases in deep learning frameworks, it highlights how graph structures present a unique challenge due to their high-dimensional, flexible nature and arbitrary connectivity. It then expands on the intractable design space of GNNs, where countless architectural variations interact intricately with graph properties, often leading to no single configuration consistently outperforming others. Addressing this, the chapter discusses how graph-specific properties can significantly influence GNN performance, sometimes overshadowing the impact of layer choice. By presenting empirical evidence supporting these claims, it underscores the pressing need for principled approaches that bridge theoretical insights with practical architecture design. Collectively, this chapter offers a *critical synthesis of the interplay between graph attributes and properties, network design, and task requirements*, following a thorough literature review, bringing forward the expected benefits of more task-aware GNN architectures—a paradigm shift with potential to advance the field of graph-based deep learning.

Matching Network Architectures and Learning Tasks by Understanding Feature/Structure Interplay Chapter 3 discloses a method for decoupling the impact of feature and structure information in the learning process. By applying this method to several synthetic graphs and real-world benchmarks, this work evidences and measures how input graph structure, feature signal, and GNN performance are intertwined. The thorough empirical analyses reveal that simple GCNs need both feature and structural information to be meaningful to learn useful node representations—if only one of these is present, GCNs cannot separate useful from meaningless information. The experiments also disclose a *sweet spot* for representation learning with GCNs, by showing that these networks excel when the feature signal is relatively weak, but the structure holds relevant information towards solving the final task under the network's requirement for local node similarity. Contrastively, we find that GNNs with more complex architectures can present a feature-preserving quality that renders them suboptimal when the sweet spot case is verified. These outcomes are turned into *empirically-driven guidelines towards matching network architectures with learning problems/tasks*—a significant contribution to the field, deeply discussed in Section 5 of the chapter.

Rethinking Design Choices for GNNs Chapter 4 addresses one of the conflicting topics within GNN research: the role of depth scaling. Its first step consists of analyzing state-of-the-art architectures and comparing them to identify the architectural requirements for deep GNNs. Following this analysis, a discussion of how depth has been evaluated is promoted, culminating in the proposition of a novel depth evaluation protocol for GNNs, that addresses and overcomes the limitations of previous methods. By implementing such a protocol, we show evidence of different types of pathological behavior in current deep GNN architectures, which manifest as redundant hidden embedding representations and seemingly useless/noisy layers. Besides the proposition, validation and interpretation of the depth evaluation protocol for GNNs, we prove that 1) the search space of GNN architectures can be narrowed down to shallow networks, contributing to both performance gains and the optimization of computational resources; 2) other architectural design choices, such as residual connections, can improve the expressivity of shallow networks, but their potential contribution is task-dependent; 3) the development of pathologies can make GNNs obliterate the usage of graph structure and ultimately behave like a Multilayer Perceptron (MLP). These conclusions are discussed in Section 5 and provide a holistic, stratified contribution towards the goal of *defining key design choices for GNN architectures.*

Guiding GNN Design Through a Principled and Task-Sensitive Approach Chapter 5 unifies the key insights developed in the previous chapters into a principled framework for informed GNN design. The chapter first distills these findings into concrete, evidence-based guidelines that help align model architecture with the specific characteristics of a given task. It then demonstrates the framework's practical relevance through a tutorial-style case study on molecular representation learning (Section 2), where a structured diagnostic process reveals and validates targeted architectural improvements. By combining theoretical understanding with empirical methodology, this chapter contributes a replicable blueprint for researchers and practitioners to design GNNs that are both efficient and problem-aware, bridging the gap between conceptual insight and real-world application.

In summary, this thesis aims to enhance the representations learned by GNNs by systematically investigating the relationship between their design choices, problem formulations, and empirical performance. By critically analyzing their practical benefits and limitations, this work seeks to bridge the gap between theoretical advancements and real-world applicability. Through rigorous empirical practices and systematic hypothesis-testing methodologies, we establish a principled framework for understanding, evaluating, and improving GNN architectures. The manuscript is closed with a conclusion chapter (Chapter 6) summarizing how the insights gained from this study not only contribute to a deeper understanding of GNNs but also provide actionable guidelines for their effective deployment across diverse domains.

2 When Graph Learning meets Deep Learning

Research in machine learning and its subfields has been positioning itself at the intersection of formal sciences and real-world applications for long [Herrmann et al., 2024]; but the balance between these two worlds can be delicate. When it comes to deep learning, the mathematical foundations can go from the fields of linear algebra and calculus to probability, statistics, and optimization theory. However, the success of deep learning in real-world scenarios is not just about mastering mathematical foundations but also about adapting them to practical constraints like scalability, interpretability, and efficiency. Graph structures, in contrast, originate from the domain of graph theory, a branch of mathematics historically focused on formal representations of relationships. While graph theory has always had practical uses, the rise of AI, big data, and computational power in the last decades has pushed it into the forefront of real-world applications like never before. *Bridging the gap between theory and practice* is what drives the real impact of AI today.

In this chapter, theoretical and practical foundations enabling deep neural networks to learn across various domains are examined, with a particular focus on graph-based learning. It explores the most relevant mathematical foundations towards understanding how to learn appropriate representations over graph structures, alongside practical strategies introduced to improve optimization mechanisms and performance. The chapter concludes with a critical discussion bridging theoretical insights with practical advancements happening at the *interface between graph and deep learning* (Section

4), establishing a solid foundation for understanding deep learning beyond Euclidean domains and into structured, relational data.

To ground the discussion, let us begin with an overview of *Graph Learning*, a field that has gained increasing attention for its ability to model complex relational data. The next section introduces fundamental concepts and methodologies used to represent and manipulate graph-structured inputs within learning frameworks. Understanding these principles is essential for appreciating how graph structures inform the design of deep learning architectures and how these architectures, in turn, enable powerful generalization over non-Euclidean domains.

1 Graph Learning

Graphs are not merely an abstract mathematical concept but serve as a fundamental tool for analyzing and interpreting intricate real-world systems. In recent decades, the proliferation of graph-structured data has expanded significantly, largely due to advancements in digital social networks, biological systems modeling, molecular chemistry, and the increasing connectivity of smart devices. The wealth of available graph data presents an enormous opportunity for scientific discovery, but fully harnessing its potential requires robust analytical techniques and computational methods. As such, an in-depth understanding of graph representations is crucial to develop machine learning algorithms that can effectively process graph-structured data.

This section outlines the fundamentals towards understanding these complex data structures, how they are represented, processed, what are their relevant properties, and how to leverage them to learn over such structures. Unlike conventional machine learning tasks that assume independent data points, graph-based learning must account for intricate interdependencies between nodes; thus, both traditional graph learning methods and deep learning models must start from an adequate representation to extract local and global patterns from graph structures. Mastering some basic mathematical foundations of graph representations is, therefore, essential for ensuring the extraction of appropriate patterns.

1.1 Graphs: Representations and Properties

A clear and formal understanding of graph representations—and of specific subclasses that simplify analysis and algorithm design—is essential for developing models that can effectively learn from structured data. To that end, we begin by establishing a rigorous definition of graph-structured data, which serves as the foundational abstraction for representing relationships between entities.

Definition 1: Graph.

A graph $\mathcal{G}$ is formally defined as an ordered pair:

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}) \tag{2.1}$$

where $\mathcal{V}$ is a finite set of nodes (vertices) and $\mathcal{E}$ is a set of edges (links), which are either unordered pairs (u, v) for undirected graphs or ordered pairs $\langle u, v \rangle$ for directed graphs, with $u, v \in \mathcal{V}$.

A particularly relevant subset of graphs is *simple graphs*, which contain at most one edge between any two nodes, have no self-loops, and solely include undirected edges, meaning that if $(u, v) \in \mathcal{E}$, then $(v, u) \in \mathcal{E}$ must also hold. Simple graphs are especially important because they provide a fundamental structure for many real-world networks while remaining mathematically and computationally manageable. Many natural networks, such as social networks, road networks, and biological interaction graphs, can be effectively modeled using simple graphs. Their undirected nature ensures mutual relationships, simplifying both interpretation and analysis. Moreover, understanding simple graphs serves as a foundation for more advanced graph structures, including directed and weighted graphs.

Graph Representations

One of the most commonly used techniques for representing a graph computationally is the *adjacency matrix*, denoted as $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ [Hamilton, 2020].

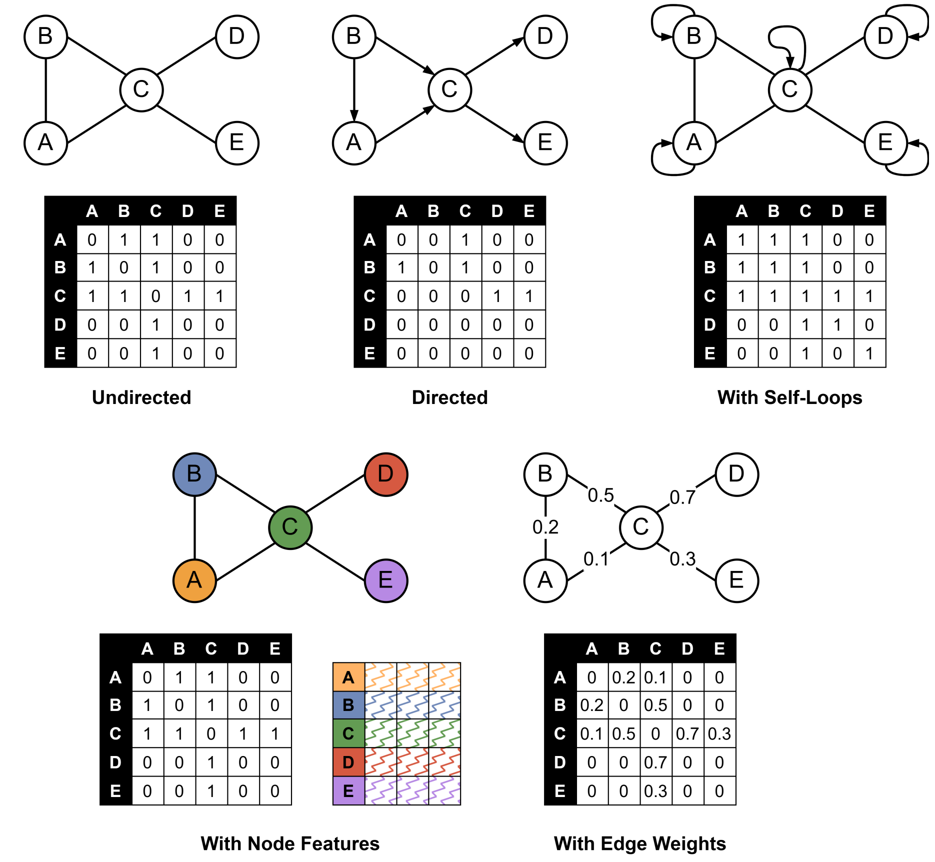
Definition 2: Adjacency Matrix.

An adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ can be formally defined by:

$$\mathbf{A}[u, v] = \begin{cases} w_{u,v}, & \text{if } (u, v) \in \mathcal{E}, \\ 0, & \text{otherwise.} \end{cases} \tag{2.2}$$

where $u, v \in \mathcal{V}$ and $w_{u,v} \in \mathbb{R}$ corresponds to the weight of edge (u, v). Unweighted graphs are a special case of this formulation where $w_{u,v} = 1 \; \forall \; (u, v) \in \mathcal{E}$.

This matrix encodes the connectivity structure of a graph by assigning each node a specific index, with entries $\mathbf{A}[u, v] = 1$ when $(u, v) \in \mathcal{E}$ and $\mathbf{A}[u, v] = 0$ otherwise. For undirected graphs, this matrix is inherently symmetric ($\mathbf{A}[u, v] = \mathbf{A}[v, u]$), whereas directed graphs may exhibit asymmetry in their adjacency representation. Additionally, some graphs contain weighted edges, in which case the adjacency matrix elements



Undirected

	A	B	C	D	E
A	0	1	1	0	0
B	1	0	1	0	0
C	1	1	0	1	1
D	0	0	1	0	0
E	0	0	1	0	0

Directed

	A	B	C	D	E
A	0	0	1	0	0
B	1	0	1	0	0
C	0	0	0	1	1
D	0	0	0	0	0
E	0	0	0	0	0

With Self-Loops

	A	B	C	D	E
A	1	1	1	0	0
B	1	1	1	0	0
C	1	1	1	1	1
D	0	0	1	1	0
E	0	0	1	0	1

With Node Features

	A	B	C	D	E
A	0	1	1	0	0
B	1	0	1	0	0
C	1	1	0	1	1
D	0	0	1	0	0
E	0	0	1	0	0

With Edge Weights

	A	B	C	D	E
A	0	0.2	0.1	0	0
B	0.2	0	0.5	0	0
C	0.1	0.5	0	0.7	0.3
D	0	0	0.7	0	0
E	0	0	0.3	0	0

Figure 2.1: Representation of several graphs in diagram and matrix forms. The exemplified undirected graph consists of a simple graph (top left).

extend beyond binary values and instead reflect real-valued weights, such as distances in transportation systems. A full definition of the adjacency matrix that accommodates this quality is given in Definition 2, where simple, unweighted graphs can be seen as a special case where all weights equal 1. Figure 2.1 provides a visual representation of these and further examples.

Beyond adjacency matrices, there are alternative graph representations that can facilitate diverse analytical applications. For example, the *edge list* format explicitly enumerates all node pairs linked by edges, providing an efficient means of storing sparse graphs. This representation is preferred by some libraries optimized for machine learning on graphs, such as PyTorch Geometric (PyG) [Fey and Lenssen, 2019], especially since most real-world graphs are sparse [Leskovec, 2023]. Another approach is the *incidence matrix*, which encodes relationships between nodes and edges in the form of a matrix $\mathbf{B} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{E}|}$, particularly useful when working with hypergraphs, i.e. graphs that allow an edge (called a hyperedge) to connect multiple nodes simultaneously.

Graphs with Attributes Graphs enriched with attribute data often require more complex representations, such as real-valued node feature matrices, edge weight tensors or even global graph-level feature vectors, to capture additional properties of entities and their interactions. A more complete definition of a graph holding all of these levels of information can be given by Equation 3. Although these representations typically rely on matrix or tensor forms that impose a specific ordering of nodes and edges, it is important to recognize that graphs themselves are inherently *unordered structures*. The identities of nodes and the order in which they appear are arbitrary; any consistent permutation of nodes and edges leaves the underlying graph unchanged. This is an important aspect that will be revisited later in the chapter, given its profound implications for the development of models that learn over graph-structured data.

Definition 3: Graph with Node, Edge and Global Features.

A graph that includes node features, edge features, and global (graph-level) features can be defined by:

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{E}^{(3)}, \mathbf{g}) \tag{2.3}$$

where:

- $\mathcal{V}$ is the **set of nodes**,
- $\mathcal{E}$ is the **set of edges**, where $(u, v) \in \mathcal{E}$ represents a connection between nodes u and v,
- $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$ is the **node feature matrix**, where each row $\mathbf{X}_u \in \mathbb{R}^d$ represents the d-dimensional feature vector of node u,
- $\mathbf{E}^{(3)} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}| \times e}$ is the **edge feature tensor**, where each entry $\mathbf{E}^{(3)}_{u,v} \in \mathbb{R}^e$ corresponds to the e-dimensional feature vector of edge (u, v),
- $\mathbf{g} \in \mathbb{R}^g$ is the g-dimensional **graph-level feature vector**, which captures global attributes of the entire graph.

Node-level features are particularly common [Hamilton, 2020] and can be represented by a matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$ that assumes an ordering of the nodes, consistent with the ordering of the adjacency matrix (as exemplified in Figure 2.1). Analogously, edges can also be associated to feature vectors, frequently stacked and represented by a 3-dimensional array $\mathbf{E}^{(3)} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}| \times e}$, thus extending the representation of discrete node connections with a new entity holding real-valued attributes. For global features, a graph-level feature vector $\mathbf{g} \in \mathbb{R}^{g}$ can be used, providing a way to encode information that is not localized to individual nodes or edges but instead describes properties of the entire graph.

More Complex Graph Structures The previous discussion focused exclusively on graphs where all nodes and edges are of a single type, i.e. *homogeneous graphs*. While homogeneous graphs effectively model many real-world systems, more complex structures can be required to capture richer relational information. For example, *heterogeneous graphs* extend this framework by incorporating multiple node and edge types, and can be represented by adding a variable to the previous graph definitions representing type-specific feature spaces. *Multiplex graphs* further generalize this concept by allowing multiple edge types between the same node pairs, making them ideal for modeling multi-layer networks such as transportation systems. Finally, *hypergraphs* capture even more intricate relationships by enabling hyperedges, which can connect arbitrary subsets of nodes instead of just pairs, allowing applications in biological interactions and higher-order social structures. Despite not being the main focus of this overview, it is important to acknowledge that these advanced graph representations are an important testament to the complexity of graph structures and provide powerful tools for modeling real-world complexity beyond traditional homogeneous graphs.

Graph Properties

Besides the fundamental representation of graph entities, additional descriptors can be defined to provide information on how nodes and edges are organized and connected within a graph. *Structural graph properties*—such as node degree, centrality, clustering, connectivity, and shortest paths—capture the underlying topology of the network and reveal how information, influence, or interactions can flow through it. They are crucial for analyzing real-world systems like social, biological, and communication networks, where structure often shapes behavior. In practice, structural properties often interact closely with *attribute-driven properties*—the features or labels associated with nodes and edges. Patterns such as homophily, attribute distributions, or label balance link the topology of the graph to its content, influencing how learning algorithms generalize, propagate information, or detect communities. Table 2.1 provides a (non-exhaustive) list of these properties and their high-level description.

Table 2.1: Structural and attribute-driven properties of graphs. These high-level descriptions illustrate the variety of commonly studied graph properties. Formal definitions of the most relevant properties for this thesis will follow.

	Property	**Description**
Structural (node-level)	(In-/Out-) Degree	Number of edges connected to a node (for directed graphs, in and out degrees can be defined).
	Degree Distribution	Statistical distribution of degrees (often follows a power law in real-world graphs).
	Degree Assortativity	Measures correlation between degrees of connected nodes.
	Connectedness	Whether all nodes are reachable from each other (undirected). If directed, whether paths exist respecting direction (strong) or ignoring it (weak).
	Eccentricity	Maximum distance from a node to all other nodes.
	Closeness Centrality	Inverse of average distance to all other nodes.
	Betweenness Centrality	Frequency a node appears on shortest paths.
	Eigenvector Centrality	Importance of a node based on neighbors' importance.
	Shortest Path	Path with the smallest total cost — usually measured in number of edges (hops) or sum of edge weights — that connects them.
	Diameter	Longest shortest path between any two nodes.
	Radius	Minimum eccentricity.
Structural (global)	(Edge) Density	Ratio of actual to possible edges.
	Clustering Coefficient	Measures triangle density.
	Laplacian Matrix	Defined by $\mathbf{L} = \mathbf{D} - \mathbf{A}$, it captures structure and is used in spectral clustering.
	Algebraic Connectivity	Second-smallest eigenvalue of Laplacian, related to connectivity.
Attribute -driven (global)	Homophily	Proportion of edges in a graph that connect nodes with the same label (class-conditional) or similar attributes (via feature similarity measures). Low homophily is often known as heterophily.
	Attribute Assortativity	Extension of degree assortativity — correlation between any node attribute across edges.
	Label Balance	Describes the proportion of different class labels in the graph (e.g., in semi-supervised learning).

Traditional machine learning techniques for graphs have long relied on structural graph properties as the foundation for feature engineering, similarity computation, and information propagation. Simple metrics like node degree and centrality are often used as input features in structure-agnostic models (e.g., logistic regression, decision trees, support vector machines). Shortest paths and other graph kernels leverage the topology of connections to compute similarity between nodes or entire graphs, also enabling structure-agnostic methods to operate on structured data [Borgwardt and Kriegel, 2005; Shervashidze et al., 2011]. Other techniques exploit the global structure of a graph through its spectral properties, which relate with measures of centrality and connectivity [Hamilton, 2020]. Meanwhile, community structure and clustering coefficients impact semi-supervised methods like label propagation, especially in the presence of homophily, where connected nodes tend to share attributes or labels [Zhu and Ghahramani, 2002; Zhou et al., 2003].

Even as the field transitions from manual feature engineering and predefined low-dimensional graph embeddings to automatic representation learning, structural and attribute-based graph properties remain fundamentally important. Contemporary approaches aim to encode graph entities into low-dimensional latent representations by training models to preserve task-relevant information, thereby eliminating the need for handcrafted intermediate features characteristic of traditional methods. Nonetheless, these models continue to operate directly on the structure of the graph and associated attributes, requiring architectures that can effectively model interactions and fully exploit both structural and attribute-driven information. The following paragraphs formalize key concepts essential to understanding the theoretical backbone of these models.

Neighborhoods and Node Degrees By defining the local context around each node, structural properties such as neighborhoods and node degrees provide the foundation for how information is aggregated and propagated across the graph. This notion of locality is essential to automatic representation learning on graphs, where models learn node and graph-level embeddings by iteratively aggregating signals from local neighborhoods.

Definition 4: Neighborhood of a node.

The neighborhood of a node $u \in \mathcal{V}$ can be given by:

$$\mathcal{N}_u = \{\, v \;:\; (u,v) \in \mathcal{E} \,\} \tag{2.4}$$

where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges.

The degree of a node can be simply described as the number of its 1-hop neighbors. Formally, this quantity can be defined as the size of the neighborhood of the node.

Definition 5: Degree of a node.

The degree of a node $u \in \mathcal{V}$ can be defined with respect to its neighborhood $\mathcal{N}_u$ as:

$$\deg(u) = |\mathcal{N}_u| \tag{2.5}$$

Degree Matrix and Graph Laplacian While the concepts of neighborhood and node degree are inherently local—defined at the level of individual nodes—their utility extends to the graph-level when expressed through appropriate mathematical representations. Notably, the degree matrix and further representations related to it, such as graph Laplacians, encapsulate information about both local node connectivity and broader structural characteristics of the graph. As such, these serve as foundational components in capturing global interaction patterns, offering a more comprehensive understanding of the topology of the graph—an attribute particularly valuable in representation learning and spectral analysis. Let us start by defining the degree matrix.

Definition 6: Degree matrix.

The degree matrix $\mathbf{D} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ of graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a diagonal matrix where:

$$\mathbf{D}[u, u] = \deg(u)\ ,\ \forall\, u \in \mathcal{V} \tag{2.6}$$

By encapsulating the connectivity information of each individual node, the *degree matrix* plays a fundamental role in characterizing the local structure of a graph. Furthermore, when combined with the adjacency matrix, it gives rise to a matrix that captures more global structural information—the *graph Laplacian*. This matrix not only reflects the connectivity of individual nodes but also encodes how pairs of nodes relate through edge structure, and can be formally defined with reference to the degree matrix and the adjacency matrix:

Definition 7: (Unnormalized) Graph Laplacian.

The Laplacian of graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is defined as:

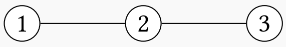
$$\mathbf{L} = \mathbf{D} - \mathbf{A} \tag{2.7}$$

where $\mathbf{D}, \mathbf{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ are the degree and adjacency matrices of $\mathcal{G}$, respectively.

One intuitive way to understand the graph Laplacian $\mathbf{L}$ is to think of it as capturing the *smoothness* of a signal defined over a graph. Imagine assigning a real-value to each node. The Laplacian measures how much these values vary across edges: if neighboring nodes have similar values, the signal is smooth and $\mathbf{L}$ produces small variations; large differences indicate high disagreement across the graph. Consider the following example:

Graph Laplacian: Illustrative Example

Consider a simple undirected graph with three nodes connected in a chain: 1—2—3. Let the signal $\mathbf{x} = [5, 4, 3]^\top$ represent a value at each node.



The degree matrix is:

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \text{and the adjacency matrix is:} \quad \mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

The unnormalized graph Laplacian is $\mathbf{L} = \mathbf{D} - \mathbf{A}$, so:

$$\mathbf{L} = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix}$$

Applying $\mathbf{L}$ to $\mathbf{x}$, we compute:

$$\mathbf{Lx} = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 4 \\ 3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

This result quantifies how much each node's value differs from the average of its neighbors. Here, node 2 is perfectly aligned with its neighbors, while nodes 1 and 3 show disagreement, highlighting areas of variation across the graph structure.

As rich representations of graph topology, graph Laplacians also inspire new analytic angles. While the adjacency matrix consists of a lossless representation of the graph, Laplacians have several mathematical properties that hold for simple graphs and naturally link to measures of graph connectivity and spectral clustering. In particular, these matrices are symmetric (i.e., $\mathbf{L}^\top = \mathbf{L}$) and positive semi-definite (i.e., $\forall\, \mathbf{x} \in \mathbb{R}^{|\mathcal{V}|}$, $\mathbf{x}^\top\mathbf{Lx} \geq 0$ always holds) [Chung, 1997]. All Laplacians also satisfy $\mathbf{L} \cdot \mathbf{1} = 0$, where

$\mathbf{1}$ represents an all-ones vector; hence zero is always an eigenvalue. Moreover, the multiplicity of the zero eigenvalue equals the number of connected components in $\mathcal{G}$ [Hamilton, 2020]. The Laplacian also satisfies a useful quadratic form (Equation 2.8) that allows the quantification of the *smoothness* of a signal $\mathbf{x}$ over the graph. Intuitively, this means that small values of $\mathbf{x}^\top \mathbf{L} \mathbf{x}$ indicate that adjacent nodes have similar values.

$$\mathbf{x}^\top \mathbf{L} \mathbf{x} = \sum_{(u,v)\in\mathcal{E}} (\mathbf{x}_u - \mathbf{x}_v)^2 \tag{2.8}$$

This smoothness perspective is foundational in *spectral graph theory*. The fact that $\mathbf{L}$ is real symmetric and positive semi-definite guarantees its eigendecomposition, given by Equation 2.9, where $\mathbf{U}$ contains orthonormal eigenvectors and $\mathbf{\Lambda}$ is a diagonal matrix of non-negative eigenvalues. These eigenvectors serve as the graph analogue of Fourier modes, enabling the decomposition of any signal $\mathbf{x}$ into "graph frequencies". Thus, spectral methods interpret graph-based signals in the frequency domain: smooth components align with low eigenvalues, while high-frequency variations (e.g., abrupt changes across edges) correspond to higher eigenvalues. This spectral view is central for spectral clustering and certain GNNs, as we will see later in the chapter.

$$\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top \tag{2.9}$$

The previous properties can also be extended to accommodate other types of graphs other than simple graphs. Furthermore, to account for degree variations across nodes, normalized versions of the graph Laplacian are also widely used. These matrices share structural similarities with the unnormalized Laplacian, but their algebraic characteristics differ slightly as a result of the normalization factors [Hamilton, 2020], which may improve numerical stability for specific applications. Two common forms are:

Definition 8: Normalized Graph Laplacians.

Considering a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with adjacency matrix $\mathbf{A}$ and degree matrix $\mathbf{D}$, the *symmetric graph Laplacian* can be given by:

$$\mathbf{L}_{\text{sym}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} \tag{2.10}$$

whereas the *random walk graph Laplacian* is given by:

$$\mathbf{L}_{\text{rw}} = \mathbf{D}^{-1}\mathbf{A} \tag{2.11}$$

Within this context, graph Laplacians emerge as one of the most expressive and versatile representations, offering a unifying framework for capturing both the structural properties and dynamics of graphs. As such, they provide a natural framework for generalizing notions of smoothness and flow to graph domains, laying the groundwork for more advanced analyses, especially those of the spectral domain.

Density Graph density provides a more coarse-grained indicator of how well a graph is connected. Its simplified formulation complements spectral connectivity measures from the graph Laplacian with an overall count of how many edges are present relative to how many could be. Intuitively, higher density implies more routes for information to flow, leading to faster mixing. In practice, metrics like degree distribution and edge density are often used together to assess connectivity, given their computational efficiency, especially in comparison with the costly eigendecomposition of the Laplacian.

Formally, the edge density d_e is defined as the ratio of present edges to the maximum possible edges of a graph. Equivalently,

Definition 9: Edge Density.

The edge density d_e of a simple graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ can be given by:

$$d_e = \frac{2 \cdot |\mathcal{E}|}{|\mathcal{V}| \cdot (|\mathcal{V}| - 1)} \tag{2.12}$$

This means that $d_e = 1$ for a complete graph and $d_e = 0$ for an edgeless graph. Directed graphs (with no self-loops) follow a different expression, $d_e = |\mathcal{E}|/(|\mathcal{V}| \cdot (|\mathcal{V}| - 1))$, since each ordered pair of distinct nodes could host a directed edge. In either case, density is a global scalar between 0 and 1 that summarizes overall connectivity.

Graphs are often informally classified as dense or sparse based on their densities. A dense graph has on the order of $O(|\mathcal{V}|^2)$ edges (so d_e close to 1), meaning nearly every pair of nodes is connected, while a sparse graph can have as little as $O(|\mathcal{V}|)$ edges. Real-world networks are typically very sparse [Leskovec, 2023].

Homophily While edge density condenses a metric of how connected the graph is, it does not describe which nodes tend to connect. The notions of homophily and heterophily describe whether connections are more likely between similar or dissimilar nodes, respectively. These concepts are important to understand how features or labels propagate across the graph and are key to designing effective graph learning algorithms.

Homophily refers to the tendency of nodes with similar features or labels to be connected. It is a pervasive structural principle in many real-world graphs—especially social and citation networks—where entities form ties with others that share common attributes [McPherson et al., 2001]. This property can facilitate smoothness in node representations, allowing some models to assume that information aggregated from the neighborhood of a node is semantically meaningful for prediction—an assumption with profound implications that we will revisit in Section 3. A common formalization of homophily is:

Definition 10: Homophily.

The homophily h of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ can be given by:

$$h = \frac{1}{|\mathcal{E}|} \sum_{(u,v)\in\mathcal{E}} \mathbb{I}[y_u = y_v], \quad \forall\,(u,v) \in \mathcal{E} \tag{2.13}$$

where y_u and y_v are the class labels of nodes u and v, $\mathbb{I}$ is the indicator function.

This formulation makes explicit that the quantity $h \in [0,1]$ captures the label homophily ratio—the proportion of edges connecting nodes of the same class.

In contrast, *heterophily* refers to a structure where edges are more likely to form between dissimilar nodes. Although less common in early benchmarks, heterophilous graphs appear frequently in biological, technological, and other settings. In such graphs, traditional GCNs may fail to propagate meaningful signals due to their reliance on local smoothing, which has been motivating the development of specialized architectures to handle heterophily explicitly attempting to decouple feature aggregation from structural proximity. This is a deeply investigated problem in GNN research, which we shall elaborate on later in this chapter.

Having established the foundational concepts and formal representations that define graph-structured data, let us now shift focus to *how these structures are leveraged in the context of machine learning*. Understanding the representational characteristics of graphs enables us to systematically explore the range of predictive tasks that can be formulated over them, each reflecting different levels of structural abstraction and learning objectives, as the following subsection will show.

1.2 Taxonomical Overview of Machine Learning Tasks on Graphs

Machine learning on graphs encompasses a diverse set of predictive tasks. These tasks can be broadly categorized based on the level of granularity at which predictions are made: node-level, edge-level, and graph-level. Each category targets a different structural unit of the graph and is characterized by distinct objectives, input-output paradigms, and downstream applications. Figure 2.2 illustrates these key differences.

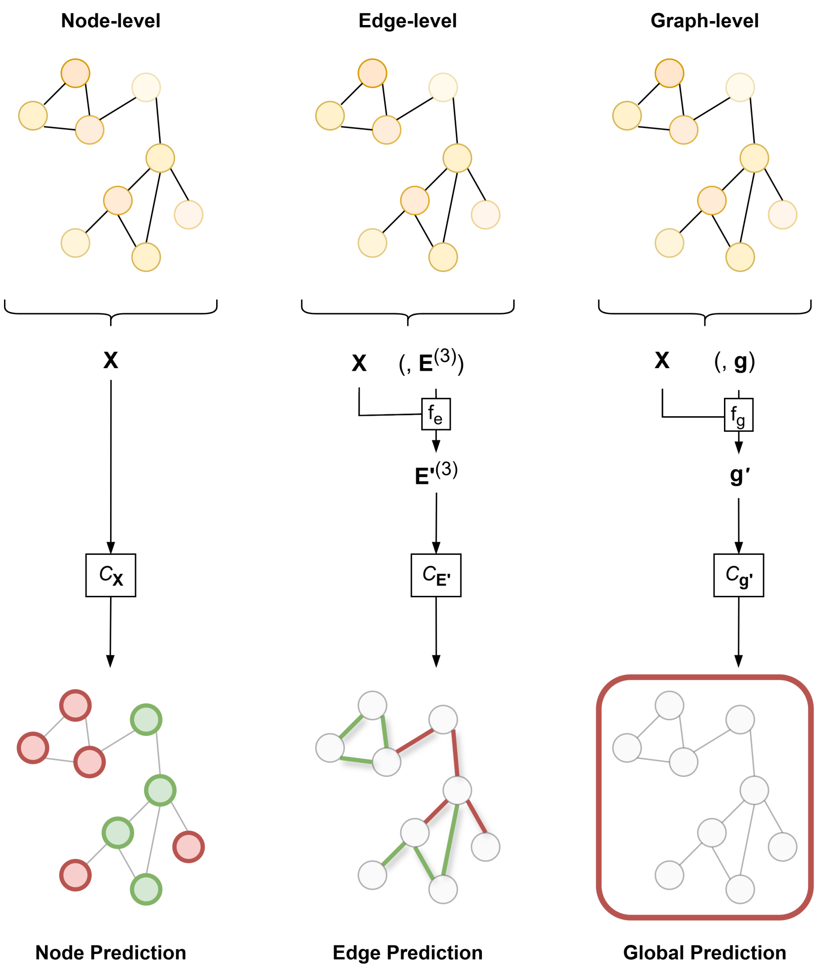


Figure 2.2: Main types of graph machine learning tasks. These tasks are defined at different levels of abstraction—node, edge, and graph—each corresponding to distinct prediction targets and learning objectives. $\mathbf{X}$, $\mathbf{E}^{(3)}$ and $\mathbf{g}$ are the node feature matrix, edge feature tensor and graph-level feature vector, respectively; f_e and f_g are pooling functions mapping node embeddings to the required final dimensions on edge and global levels, respectively; $C_{\mathbf{X}}$, $C_{\mathbf{E}'}$ and $C_{\mathbf{g}'}$ are classifiers delivering the final predictions on each of the respective levels.

Node Predictions At the *node-level*, the objective is to infer attributes or labels for individual nodes within a graph. This form of prediction is particularly relevant in settings where nodes represent entities with partially observed or unknown characteristics, such as users in a social network or documents in a citation graph. Node classification, perhaps the most standard task at this level, entails learning a function that maps each node and its local graph context to a discrete label. Mechanisms that iteratively aggregate feature information from neighboring nodes are common in this context [Kipf and Welling, 2017; Hamilton et al., 2017; Veličković et al., 2018] as these architectures enable models to learn expressive node representations that incorporate both node features and topological context, a property shown to be critical for robust performance on sparse and noisy graphs. Importantly, node-level learning is typically formulated in a semi-supervised setting, where only a subset of nodes is labeled, thus necessitating models that can effectively propagate supervision signals through the graph structure.

Edge Predictions *Edge-level* tasks, in contrast, focus on the relationships between nodes and are particularly concerned with predicting the presence, absence, or type of an edge. A common example is link prediction, which aims to estimate the probability that a connection exists between two given nodes, based on their embeddings and potentially on attributes associated with the edge itself. This task underpins several practical applications, including friend recommendation in social networks and protein interaction discovery in biological networks. Edge-level learning often involves constructing a latent space in which the embeddings of related node pairs are combined to create or extend edge-level embeddings. This can be achieved via simple methods, such as concatenation or sum, but also more complex ones, such as neural networks.

Global Predictions Another category encompasses *graph-level* tasks, where the goal is to derive a holistic representation of an entire graph for the purpose of classification or regression. These tasks are essential in domains such as cheminformatics and bioinformatics, where each graph may represent a molecule, and the objective is to predict properties like toxicity or solubility. Unlike node- and edge-level tasks, graph-level prediction requires an effective graph pooling function—a mechanism to pool variable-sized sets of node embeddings into a fixed-dimensional vector (illustrated by f_{g} in Figure 2.2). Early approaches relied on simple permutation-invariant operations such as summation or averaging, but more sophisticated techniques, including hierarchical pooling methods (e.g. DiffPool [Ying et al., 2018b]), have been developed to capture multiscale graph features. The expressiveness of graph-level models is deeply linked to their capacity to preserve both local structural patterns and global topological properties, a challenge that continues to motivate research in the design of powerful and efficient graph neural architectures.

Subgraph Predictions While most graph learning tasks are formulated at the node, edge, or whole-graph level, an increasingly relevant direction focuses on *subgraph-* or *community-level* tasks, where the objective is to reason about localized regions of the graph that correspond to functionally or structurally coherent groups of nodes. These tasks include community detection, motif classification, and localized anomaly detection, and are particularly pertinent in settings where modularity or regional patterns are semantically meaningful, such as protein complexes in biological networks or subcircuits in electronic design.

Overall, this taxonomy of machine learning tasks on graphs reflects the diversity of problems that arise when modeling structured data. Each task level not only corresponds to different types of prediction but also necessitates distinct computational paradigms and inductive biases. As graph representation learning continues to mature, the unifying challenge remains to develop models that are simultaneously expressive, scalable, and capable of leveraging the rich relational structure intrinsic to graph data on all levels.

To design such models, it is crucial to consider not only the task-specific requirements but also the structural properties inherent to graph data. In particular, the *absence of a canonical node ordering and the presence of complex relational patterns* call for principled approaches that can generalize across different graph topologies. This motivates the need for inductive biases that capture symmetries and invariances in graph-structured domains. These strategies play a central role in shaping the design and behavior of modern graph learning algorithms, as the following subsection will demonstrate in detail.

1.3 Geometric Priors in Graph Machine Learning

Graph-based machine learning involves the analysis of data represented as graphs, which are inherently non-Euclidean and combinatorial structures. Unlike data in vector spaces, graphs lack a natural ordering of their elements and exhibit rich symmetries under permutations of node identities. *Geometric priors* are inductive biases that encode assumptions about these symmetries and invariances, guiding the design of learning algorithms that operate on graphs [Bronstein et al., 2021]. They ensure consistency across structurally equivalent representations and improve the generalizability of learned representations. Let us dive into how to define these graph symmetries and which type of functions can appropriately encode them.

Graph Symmetries

In their work on geometric deep learning, Bronstein et al. [2021] define symmetries as, quoting, "structure preserving and invertible maps *from an object to itself*". These maps can also be called *automorphisms*. Thus, particularly for graphs, automorphisms describe the ways a graph can be mapped onto itself while preserving its topology. This means that graph automorphisms capture the intrinsic structural redundancies of a graph: nodes can be relabeled (permuted) without altering the edge connectivity. Since node labels in most graph-based tasks are arbitrary, models should remain invariant under these automorphisms to avoid encoding spurious dependencies on the node ordering [Xu et al., 2019]. More formally,

Definition 11: Graph Automorphisms.

An automorphism of graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a permutation $\pi : \mathcal{V} \to \mathcal{V}$ of its nodes that preserves adjacency. Thus, $\forall u, v \in \mathcal{V}$,

$$\mathbf{A}[u, v] = \mathbf{A}[\pi(u), \pi(v)] \tag{2.14}$$

If expressed in terms of the adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$, this means that a permutation matrix $\mathbf{P}$ represents an automorphism if and only if

$$\mathbf{P}^\top \mathbf{A} \mathbf{P} = \mathbf{A} \tag{2.15}$$

The set of all such permutations forms the *automorphism group* $\mathrm{Aut}(\mathcal{G})$ of the graph.

This group-theoretic view underlies many geometric priors in graph learning. For instance, any GNN or graph kernel that respects these symmetries will treat automorphic nodes identically, ensuring consistent representation of symmetric structures. In model design, exploiting graph symmetries leads to architectures that are *equivariant* or *invariant* under the action of $\mathrm{Aut}(\mathcal{G})$. For example, graph convolution or message-passing layers aggregate neighbor information without bias toward specific node labels, implicitly respecting these symmetries. Likewise, the role of symmetries appears in graph kernel methods: graph kernels typically compute signatures (e.g. based on substructures or counts) that are invariant to any automorphism of the input. In all cases, the automorphism group of $\mathcal{G}$ defines the inductive bias: invariance to these permutations ensures that the model's behavior depends only on the structure of the graph, not on arbitrary indexing of nodes.

Graph Isomorphism

Graph isomorphism formalizes the notion of equivalence for two different graphs, i.e., when two graphs have the same connectivity but a different labeling of nodes. In the language of priors, any predictive function on graphs should ideally assign the same output to isomorphic graphs: this is the fundamental invariance that graph-based models must respect. As a related concept, graph automorphisms can be understood as isomorphisms from a graph onto itself, capturing the intrinsic symmetries of the graph under node relabeling, as previously discussed.

Definition 12: Graph Isomorphism.

Two graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and $\mathcal{H} = (\mathcal{V}', \mathcal{E}')$ are isomorphic ($\mathcal{G} \cong \mathcal{H}$), if their adjacency matrices $\mathbf{A}_{\mathcal{G}}$ and $\mathbf{A}_{\mathcal{H}}$ satisfy:

$$\mathbf{A}_{\mathcal{G}} = \mathbf{P}^{\top}\mathbf{A}_{\mathcal{H}}\mathbf{P} \tag{2.16}$$

for a permutation matrix $\mathbf{P}$.

In practical terms, isomorphic graphs are indistinguishable by any invariant graph feature or descriptor. This concept is central to the design and analysis of graph learning methods, as it guarantees that isomorphic graphs map to the same low-dimensional embeddings. Analogously, it allows us to define yet another desirable quality for learning models: non-isomorphic graphs should map to different low-dimensional embeddings. This quality, while intuitive, can be difficult to achieve. For example, Xu et al. [2019] and Morris et al. [2019] rigorously demonstrated that message-passing GNNs possess inherent limitations in distinguishing certain classes of non-isomorphic graphs. Specifically, in the absence of additional information, some GNNs may map structurally different graphs to identical representations due to insufficient discriminative power. This theoretical insight has motivated the development of more expressive architectures that can capture richer structural nuances—such as GIN [Xu et al., 2019]—which we will revisit later in the chapter.

Permutation Invariant and Equivariant Functions

This context is essential to define key inductive biases for graph learning models: when operating on graph-structured data, it is crucial that functions are consistent under permutations of node ordering. This requirement is formalized through permutation invariance and equivariance, expressed in terms of how node features $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$ and the adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ transform under a permutation $\mathbf{P} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$, where $\mathbf{P}$ is a permutation matrix.

A graph-level function $f(\mathbf{X}, \mathbf{A})$ is said to be *permutation invariant* if Equation 2.17 holds for any permutation matrix $\mathbf{P}$. This implies that the output of f remains unchanged regardless of how nodes are relabeled.

$$f(\mathbf{PX}, \mathbf{PAP}^{\top}) = f(\mathbf{X}, \mathbf{A}) \tag{2.17}$$

Similarly, we can say that a node-level function $\mathbf{F}(\mathbf{X}, \mathbf{A})$ is *permutation equivariant* if $\mathbf{F}$ satisfies Equation 2.18, meaning that permuting the nodes of the input graph results in a correspondingly permuted output.

$$\mathbf{F}(\mathbf{P}\mathbf{X}, \mathbf{P}\mathbf{A}\mathbf{P}^\top) = \mathbf{P}\mathbf{F}(\mathbf{X}, \mathbf{A}) \tag{2.18}$$

To ensure practical expressivity and computational efficiency, permutation equivariant functions are often constructed in a *local* manner. That is, the output for each node depends only on its features and those of its immediate neighbors. A general recipe for constructing permutation equivariant functions involves defining a shared, permutation-invariant function ϕ that operates on a node's feature vector, $\mathbf{x}_u$, and its neighborhood multiset $\mathbf{X}_{\mathcal{N}_u} = \{\{\mathbf{x}_v \mid v \in \mathcal{N}_u\}\}$:

$$\mathbf{F}(\mathbf{X}, \mathbf{A}) = \begin{bmatrix} \phi(\mathbf{x}_1, \mathbf{X}_{\mathcal{N}_1}) \\ \phi(\mathbf{x}_2, \mathbf{X}_{\mathcal{N}_2}) \\ \vdots \\ \phi(\mathbf{x}_n, \mathbf{X}_{\mathcal{N}_n}) \end{bmatrix} \tag{2.19}$$

Since the function ϕ is shared across all nodes and is invariant to the ordering of the neighborhood features, the resulting function $\mathbf{F}$ is permutation equivariant by construction [Bronstein et al., 2021]. When ϕ is injective, this construction yields a representation update that preserves structural distinctions between a broad class of non-identical graphs [Xu et al., 2019], thereby aligning the model's expressive capacity with established refinement principles from graph theory.

To complete our overview of Graph Learning, we have highlighted the central role of geometric priors—model-agnostic principles that ensure consistency and generalizability across structurally equivalent or dissimilar graphs. These priors are foundational to a broad range of learning paradigms and serve as a unifying lens through which to understand learning over relational data.

Building on this foundation, the next section turns to the core principles of *Deep Learning*, with the aim of uncovering how its process of creating hierarchical representations, along with its optimization strategies, can be brought into dialogue with graph-structured data. This transition sets the stage for exploring how the two paradigms—graph learning and deep learning—interact and reinforce each other in the development of expressive, structure-aware learning models.

2 Deep Learning

Deep learning is a subfield of machine learning that employs artificial neural networks composed of multiple, sequential layers to form deep architectures capable of modeling complex patterns and relationships in data. These networks acquire hierarchical feature representations from raw inputs through non-linear transformations, guided by a training process based on optimization techniques such as gradient descent.

A central theoretical principle of deep learning is that sufficiently deep neural networks possess the capacity to approximate any measurable function, provided they have adequate width and appropriate non-linearities—a result formalized by the Universal Approximation Theorem [Hornik et al., 1989]. Nevertheless, this theoretical capacity does not guarantee practical trainability, generalization, or computational efficiency. To mitigate the limitations of generic multilayer perceptrons, modern architectures integrate inductive biases that align with the underlying structure of the data. These biases may be geometric and domain-specific—exemplified by convolutional, recurrent, and graph-based models—or more generic training facilitators, such as residual connections, normalization layers, and attention mechanisms.

This section explores the theoretical foundations of universal approximation, the motivation for incorporating inductive biases (especially from a geometric perspective), and the range of architectural techniques that have emerged to stabilize training and improve generalization performance in practice. The discussion begins with an introduction to the foundational components of artificial neural networks, including perceptrons, objective functions, and gradient-based optimization. The Universal Approximation Theorem is then discussed alongside its limitations in practical settings, motivating the need for inductive biases that exploit the structure of specific domains. Subsections 2.3 and 2.4 outline the methods that leverage geometric inductive biases and broader design principles that act as training facilitators, respectively.

2.1 Artificial Neural Networks

Artificial neural networks (ANNs) are composed of layers of interconnected processing units called *neurons*, organized to transform input data through successive linear and non-linear operations. Their fundamental unit is the *perceptron*, introduced by Rosenblatt [1958], operating as a weighted linear transformation followed by a non-linear activation. As such, for a batch of B input samples $\mathbf{X} \in \mathbb{R}^{B \times d_{in}}$, the output of a single layer is given by Equation 2.20, where $\mathbf{W} \in \mathbb{R}^{d_{in} \times d_{out}}$ and $\mathbf{b} \in \mathbb{R}^{1 \times d_{out}}$ are the weight matrix and bias vector, respectively, and σ is a non-linear activation function applied element-wise (e.g., ReLU, sigmoid, or tanh).

$$\hat{\mathbf{Y}} = \sigma(\mathbf{X} \cdot \mathbf{W} + \mathbf{b}) \tag{2.20}$$

Stacking multiple perceptrons into L layers results in a *Multilayer Perceptron* (MLP), which generalizes this formulation to:

$$\mathbf{H}^{(0)} = \mathbf{X}, \quad \mathbf{H}^{(l+1)} = \sigma(\mathbf{H}^{(l)}\mathbf{W}^{(l+1)} + \mathbf{b}^{(l+1)}), \quad l = 0, \dots, L-1 \tag{2.21}$$

The output of the final layer $\mathbf{H}^{(L)} \in \mathbb{R}^{B \times d_L}$ corresponds to either classification logits or regression predictions, which can be represented by $\hat{\mathbf{Y}}$. In other words, this neural network with L layers implements a parametric function $f_\theta(\mathbf{X})$ where $\theta = \{\mathbf{W}^{(l)}, \mathbf{b}^{(l)}\}_{l=1}^{L}$ are the learnable parameters (Equation 2.22).

$$\mathbf{H}^{(L)} = \hat{\mathbf{Y}} = f_\theta(\mathbf{X}) = \sigma^{(L)}\left(\cdots \sigma^{(2)}\left(\sigma^{(1)}\left(\mathbf{H}^{(0)}\mathbf{W}^{(1)} + \mathbf{b}^{(1)}\right)\cdots\right)\right) \tag{2.22}$$

Objective Function and Optimization Training a neural network involves adjusting θ to minimize a task-specific objective function $\mathcal{L}(\theta)$, often called the *loss function*, which measures the discrepancy between predicted outputs $\hat{\mathbf{Y}} = f_\theta(\mathbf{X})$ and true targets $\mathbf{Y}$ [Rumelhart et al., 1986]. Common loss functions include the mean squared error (MSE) for regression (Equation 2.23) and cross-entropy (CE) for classification (Equation 2.24), where N is the number of samples and C is the number of classes of the classification problem.

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(\hat{\mathbf{Y}}_i - \mathbf{Y}_i)^2 \tag{2.23}$$

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C}\mathbf{Y}_{ij}\log\hat{\mathbf{Y}}_{ij} \tag{2.24}$$

Model parameters θ are updated via *gradient descent*, an iterative optimization algorithm that takes steps in the direction of the negative gradient of the loss, according to Equation 2.25, where $\eta > 0$ is the learning rate and $\nabla_\theta\mathcal{L}(\theta)$ denotes the gradient of the loss with respect to the parameters.

$$\theta \leftarrow \theta - \eta\nabla_\theta\mathcal{L}(\theta) \tag{2.25}$$

In practice, stochastic gradient descent (SGD) or its variants (e.g., Adam [Kingma and Ba, 2015]) are used to further adapt the update dynamics for better convergence.

2.2 Universal Approximation Theorem

A foundational theoretical result in deep learning is the *Universal Approximation Theorem* (UAT), which states that feedforward neural networks, specifically MLPs with at least one hidden layer containing a sufficient number of neurons and suitable non-linear activation functions, can approximate any continuous function on a compact subset of $\mathbb{R}^n$ to arbitrary precision [Cybenko, 1989; Hornik et al., 1989].

Theorem 1: Universal Approximation Theorem.

For a continuous function $f : \mathbb{R}^n \to \mathbb{R}$ and any $\varepsilon > 0$, there exists an MLP g such that:

$$\sup_{x \in K} |f(x) - g(x)| < \varepsilon$$

for every compact subset $K \subset \mathbb{R}^n$.

Theorem 1 formalizes this idea, establishing that neural networks are universal function approximators. This implies that neural networks are capable, in principle, of representing any arbitrarily complex function, provided that they have enough capacity, which is an important proof of the power of this class of methods.

However, the UAT does not imply that such functions are *trainable* in practice. The theorem is existential—it guarantees the existence of parameters that approximate a target function but provides no insight into whether these parameters can be found efficiently via gradient-based optimization. In fact, optimization becomes increasingly more complex with dimensionality. In high-dimensional spaces, the optimization landscape is often ill-conditioned (i.e., small changes in the model parameters can lead to large and unpredictable changes in the loss value) and characterized by numerous saddle points, where the gradient is zero (i.e., vanishes) even if the point is not a local minimum or maximum [Dauphin et al., 2014]. As networks become deeper, they also become harder to train due to vanishing gradients and increasingly complex loss surfaces [Hochreiter, 1998]. Moreover, as the input dimensionality increases, the underlying function to be learned typically spans a more complex and fragmented surface, with variations occurring along many axes. This proliferation of degrees of freedom makes accurate approximation increasingly challenging, an expression of the broader phenomenon known as the curse of dimensionality [Bellman and Kalaba, 1959]. Let us break down what this means for deep networks when applied in practical settings.

The Curse of Dimensionality

The *curse of dimensionality* describes how the volume of the input space grows exponentially with its dimensionality, which presents a significant barrier to the practical realizability of universal function approximation. In high-dimensional settings, sampling the input space densely becomes intractable, requiring an exponential increase in training data to maintain the same coverage or resolution. As a result, MLPs trained in such regimes often overfit sparse data or fail to converge to meaningful solutions. Furthermore, even though MLPs are universal in theory, achieving accurate approximations for high-dimensional functions often demands an infeasibly large model and training iterations [Barron, 1993].

This curse manifests not only in sample complexity but also in optimization dynamics. High-dimensional input spaces can induce highly non-convex error surfaces, leading to poor gradient signals, plateaus, or sharp minima during training. These issues are exacerbated when using generic fully connected architectures that treat each input dimension as statistically independent, failing to exploit the inherent structure often present in data from real-world domains such as images, sequences, or graphs.

To overcome these challenges, modern deep learning architectures increasingly incorporate *geometric inductive biases*—structural constraints that reflect known invariances or locality properties in the data. These geometric constraints reduce the effective dimensionality of the learning problem by limiting the space of functions a network can express to those consistent with the known symmetries of the domain. By doing so, they mitigate the curse of dimensionality and facilitate more data-efficient learning, improved generalization, and more stable optimization.

Hence, while the UAT affirms that MLPs are theoretically sufficient for function approximation, practical considerations—especially in high-dimensional regimes, where most relevant real-world problems fit—necessitate the *introduction of architectures that embed prior knowledge through geometric structure.* The next section discusses these domain-specific inductive biases, formalizing their role in overcoming the limitations of naive MLP architectures and bridging the gap between theoretical expressiveness and practical trainability that enables modern neural networks to learn complex functions efficiently and robustly across diverse domains.

2.3 Geometric Inductive Biases

A powerful approach to improving neural network performance is to incorporate inductive biases that encode prior knowledge about the structure of the input domain. Among the most effective of these are geometric inductive biases, which exploit regularities such as translation invariance in images or temporal order in sequences. These architectural constraints serve to reduce the hypothesis space by limiting it to functions that respect specific symmetries in the data, thereby improving both generalization and sample efficiency.

The geometric structure of many data domains can be formalized using concepts from group theory [Bronstein et al., 2021], which provides a principled framework for describing symmetry transformations. Neural network architectures that incorporate these symmetries embed this prior knowledge directly into their design. This alignment between model structure and domain geometry allows for more efficient learning by biasing the model towards solutions that are inherently compatible with the known regularities of the data.

Group Symmetries Imposing constraints on learning models can be framed through the lens of group theory, by enforcing appropriate symmetry properties under the action of a group G. Let G act on input space $\mathcal{X}$ and the output space $\mathcal{Y}$ via $g \cdot x$ and $g \cdot y$, respectively. A function $f : \mathcal{X} \to \mathcal{Y}$ is said to be:

- *Invariant* if $f(g \cdot x) = f(x), \quad \forall\, g \in G$, meaning that transformations of the input under G leave the output unchanged.
- *Equivariant* if $f(g \cdot x) = g \cdot f(x), \quad \forall g \in G$, meaning that applying a transformation to the input corresponds to an equivalent transformation of the output.

By enforcing invariance or equivariance, one reduces the hypothesis space of f to functions that respect the symmetries of the domain, thereby encoding strong inductive biases that can enhance generalization [Bronstein et al., 2021].

Translation Symmetry in Images Image data is typically organized on a regular grid, such as pixels in a two-dimensional plane $\mathbb{Z}^2$. A key property of many visual patterns is that they remain recognizable even when they are shifted spatially—for example, a cat in the top-left corner of an image is still a cat if moved to the bottom-right. This property, known as translation symmetry, is exploited by Convolutional Neural Networks (CNNs).

CNNs are designed to be equivariant to translations, meaning that if an input image is shifted, the resulting output (such as a feature map) shifts in the same way. This is achieved through the use of convolutional layers, which apply the same set of learnable filters/weights across all locations in the image. This operation is equivariant to translations s belonging to the discrete translation group $S = \mathbb{Z}^2$, acting on input space $\mathcal{X}$ by shifting the spatial indices of the image pixels. As such, shifting an input x by s results in a corresponding shift of the output $f(s \cdot x) = s \cdot f(x), \ \forall s \in \mathbb{Z}^2$. In other words, the convolution filters scan the image using a sliding window, detecting local features—such as edges or textures—independently of their position. For instance, a filter that detects horizontal edges will respond similarly whether the edge appears at the top, center, or bottom of the image.

By exploiting translation equivariance, CNNs achieve parameter sharing, significantly reducing the number of learnable parameters compared to fully connected architectures. This makes them particularly well-suited for visual tasks where local patterns repeat across space, such as image classification, object detection, and segmentation [Lecun et al., 1998; Goodfellow et al., 2016].

Temporal Symmetry in Sequences Sequential data, such as time series or natural language, exhibit temporal symmetry: patterns can occur at any point in time and still carry the same meaning. For example, a particular word in a sentence or a spike in a sensor reading has the same significance regardless of its position in the sequence. This can be modeled as an invariance under shifts indexed by the natural numbers $\mathbb{N}$.

Recurrent Neural Networks (RNNs) incorporate this principle by applying the same set of weights recurrently across time steps [Elman, 1990]. At each time step t, the RNN updates a hidden state h_t based on the current input x_t and the previous hidden state h_{t-1} using a fixed transformation given by Equation 2.26, where σ is a non-linear activation function. Because the transformation parameters $\mathbf{W}_h, \mathbf{W}_x$ are shared across all time steps, RNNs are equivariant to temporal shifts: a repeated pattern at different time positions will lead to similar hidden state activations, enabling the network to generalize across time.

$$\mathbf{h}_t = \sigma(\mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{W}_x \mathbf{x}_t + \mathbf{b}) \tag{2.26}$$

Long Short-Term Memory (LSTM) networks [Hochreiter and Schmidhuber, 1997] extend RNNs with gating mechanisms to better capture long-range dependencies, while mitigating vanishing gradient problems. Crucially, they preserve the same temporal inductive bias by applying shared operations across time steps, improving trainability without altering the underlying symmetry. As such, both architectures apply a shared transformation at every time step, enabling them to consistently interpret shifted patterns in a sequence, making them well-suited for tasks such as language modeling, speech recognition, and time series forecasting, where the temporal order of information is critical but patterns may occur at varying positions within the input.

Permutation Symmetry in Graphs Graphs exhibit a distinct form of symmetry: the identities of nodes are arbitrary, and any relabeling that preserves edge connectivity—i.e., a graph automorphism—should not affect the output of a model. This reflects a permutation symmetry over node indices. As discussed earlier in the chapter (Section 1.3), effective graph learning models must be robust to such symmetries to avoid encoding spurious dependencies on node ordering.

Graph Neural Networks (GNNs) achieve this by employing local message-passing mechanisms, where each node updates its representation through a shared function applied over its neighbors' features. Since these neighborhood aggregations are typically defined via permutation-invariant operations, such as summation or mean, the resulting update functions are equivariant to permutations of node labels (see Section 1.3). Consequently, if the nodes of the input graph are permuted, the output node representations are permuted in the same way. This ensures that the learned representations depend solely on the graph structure rather than arbitrary node indexing. Such permutation symmetry is essential for generalization across isomorphic graphs and underpins the expressive power of many GNN variants [Bronstein et al., 2021; Xu et al., 2019].

These architectural paradigms—CNNs, RNNs/LSTMs, and GNNs—demonstrate how different group symmetries can be operationalized as geometric inductive biases. They constrain the hypothesis space in a manner consistent with the data's underlying geometric or relational structure, thus enabling more efficient and generalizable learning. The next section shifts focus to complementary strategies that *improve learning dynamics without assuming geometric priors.*

2.4 Beyond Geometry: Techniques for Efficient Training

Despite the mitigation offered by geometric inductive biases, significant challenges persist in training deep networks efficiently and effectively. In particular, issues such as vanishing or exploding gradients, poor optimization landscapes, unstable convergence, and overfitting are not always resolved by architectural alignment with domain geometry. It is in this context that non-geometric *training facilitators* become essential components of modern deep learning systems. These mechanisms—including residual connections, normalization layers, attention mechanisms, initialization schemes, and regularization techniques—serve to improve trainability, accelerate convergence, and enhance generalization in ways that are often orthogonal to geometric structure.

Crucially, non-geometric facilitators do not rely on specific assumptions about the symmetry or topology of the data domain. Instead, they function by shaping the optimization dynamics or the capacity control of the network, addressing structural inefficiencies and optimization limitations inherent in generic MLPs and even in geometrically-informed networks. Their widespread adoption reflects a dual approach in deep learning architecture design: while geometric biases constrain the hypothesis space in line with domain-specific symmetries, non-geometric mechanisms stabilize and guide the learning process across a wide range of domains and tasks. This interplay between inductive biases and training facilitators forms the backbone of contemporary deep learning systems, as will be further explored in the subsequent subsections.

Residual Connections. Residual connections, introduced by ResNets [He et al., 2016], use identity mappings to add the input of a layer to its output, formulated as Equation 2.27.

$$\mathbf{x}^{(l+1)} = \mathbf{x}^{(l)} + f_\theta(\mathbf{x}^{(l)}) \tag{2.27}$$

This simple yet powerful design alleviates the vanishing gradient problem by preserving signal propagation through deep networks, enabling stable optimization and iterative feature refinement. Residual blocks encourage learning small perturbations around the identity function rather than entirely new representations, improving convergence and promoting feature reuse [Veit et al., 2016; Zhang et al., 2019].

Variants of residual connections extend their utility across architectures. Early feature injection strategies (e.g., initial residual) reinforce initial signal propagation by reintroducing inputs at multiple depths [He et al., 2016], while multi-scale and layer-wise aggregation methods combine representations across layers to capture diverse feature hierarchies and mitigate over-smoothing [Huang et al., 2017]. Such approaches enhance expressivity and stability in deep or hierarchical models, including convolutional, recurrent, and graph-based networks.

Despite their success, residual connections face limitations. Their effectiveness depends on factors like depth, initialization, and normalization [Zhang et al., 2019], and they may be underutilized, leading to redundancy or insufficient transformation [Veit et al., 2016]. These challenges have motivated refinements—such as reparameterizations and architectural variations—to improve learning dynamics and representational diversity [Brock et al., 2021].

Attention Mechanisms Attention mechanisms enable models to dynamically weight input elements based on relevance, improving long-range dependency modeling. Introduced in RNN-based machine translation [Luong et al., 2015; Bahdanau et al., 2016], early attention alleviated fixed-length bottlenecks and enabled soft alignment, but was tightly coupled to sequential computation and offered limited interpretability beyond structured tasks [Jain and Wallace, 2019].

Self-attention, popularized by Transformers [Vaswani et al., 2017], generalizes attention via pairwise token interactions, enabling full parallelism and flexible context modeling. However, it lacks strong inductive biases like locality [Cordonnier et al., 2020], scales quadratically with input size [Wang et al., 2020], and produces weights that may not reflect meaningful explanations [Jain and Wallace, 2019].

These limitations have driven hybrid approaches that combine attention with structural priors to improve sample efficiency, scalability, and generalization. In particular, attention has also been applied for graph data in the context of GNNs (e.g., GATs [Veličković et al., 2018]), where it facilitates adaptive neighborhood aggregation. Yet, similar concerns arise regarding its interpretability and robustness [Hu et al., 2025].

Normalization Layers Normalization techniques mitigate internal covariate shift by standardizing activations, promoting stable gradients and faster convergence. Batch Normalization (BN) [Ioffe and Szegedy, 2015] performs normalizations across mini-batches, enabling higher learning rates and improved training speed. Following the introduction of BN, alternative methods such as LayerNorm [Ba et al., 2016], InstanceNorm [Ulyanov et al., 2017], and GroupNorm [Wu and He, 2018] were also proposed to adapt normalization to different scopes—from individual features to groups of channels. When appropriately used, normalization methods can be powerful ways of improving training stability, optimization efficiency and generalization of deep networks [Huang, 2022].

Initialization Schemes Initialization strategies were designed to prevent vanishing or exploding gradients by maintaining variance across layers through techniques like Xavier [Glorot and Bengio, 2010] and He [He et al., 2015], which tailor weight variance to specific activation functions. Orthogonal initialization [Saxe et al., 2014] was also proposed to maintain stable signal propagation across layers, by preserving the norm of input vectors during forward and backward passes. Despite their past success, these classical initialization assumptions have become less critical as models grow larger, training durations increase, and complementary components like normalization, residual connections, and adaptive optimizers [Kingma and Ba, 2015] mitigate many of the optimization difficulties that proper initialization was originally designed to address.

Regularization Techniques Regularization techniques reduce overfitting by constraining model capacity or altering training dynamics. Dropout [Srivastava et al., 2014] stochastically deactivates neurons during training to encourage redundancy and improve generalization. Weight decay applies ℓ_2 penalties on parameters, early stopping halts training before overfitting, and data augmentation expands the training data. Despite their effectiveness, these methods come with caveats. For example, conventional dropout has been deemed unnecessary for GNNs [You et al., 2020], while spatial variants of dropout can be beneficial for graph data [Rong et al., 2020]. In general, excessive regularization also risks reinforcing dataset biases, decreasing generalization [Murel and Kavlakoglu, 2023].

To conclude our overview of Deep Learning, we have examined the core principles that make neural networks effective tools for hierarchical representation learning, including their expressive capacity, training dynamics, and architectural innovations. These insights complement the structural foundations introduced in the context of Graph Learning, highlighting both the strengths and limitations of deep models when applied to non-Euclidean domains. With these two pillars in place, we are now prepared to explore their intersection: the field of *Graph Neural Networks* (GNNs). GNNs emerge from the synthesis of relational inductive biases and deep function approximation, offering a principled framework for learning over graph-structured data by integrating ideas from both domains.

3 Graph Neural Networks

GNNs have emerged as a powerful framework for learning from graph-structured data, unifying the strengths of traditional graph algorithms and modern deep learning techniques. Their capacity to capture complex relational patterns and represent multi-scale dependencies has significantly advanced node classification, link prediction, and graph-level prediction tasks.

This section expands on how deep learning techniques have been adapted to process graph-structured data. It explores the evolution from spectral graph convolutions to spatial methods, the role of message-passing frameworks, and architectural design considerations, including works that systematically explore the GNN design space [You et al., 2020]. Finally, this section discusses the learning challenges inherent to GNNs, such as oversmoothing, oversquashing, and harmful heterophily, highlighting current research efforts to mitigate these issues. This collection provides a comprehensive understanding of how graph learning meets deep learning, positioning GNNs as a pivotal class of models in modern machine learning on relational data.

3.1 Graph Convolution: From Spectral to Spatial

Spectral graph theory provides a rigorous foundation for analyzing signals defined over graphs, making it a natural starting point for introducing the concept of *graph convolution*. As previously discussed in Subsection 1.1, the graph Laplacian encapsulates the connectivity of a graph and enables us to measure the *smoothness* of a signal $\mathbf{x}$—a real-valued vector defined over nodes. Such a signal might represent, for instance, temperature across a sensor network, influence in a social network, or any scalar quantity associated with each node.

Through the spectral decomposition of the Laplacian, $\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^\top$, one obtains the graph Fourier basis $\mathbf{U}$ and its associated eigenvalues Λ (recall the intuition of "graph frequencies" discussed for Equation 2.9 in Subsection 1.1). This allows any graph signal $\mathbf{x} \in \mathbb{R}^{|\mathcal{V}|}$ to be expressed in the spectral domain as $\hat{\mathbf{x}} = \mathbf{U}^\top\mathbf{x}$. In this framework, a graph convolution is defined in the spectral domain by Equation 2.28, where $g(\Lambda)$ is a diagonal filter function applied to the graph frequencies.

$$g(\mathbf{L})\mathbf{x} = \mathbf{U}g(\Lambda)\mathbf{U}^\top\mathbf{x} \tag{2.28}$$

While this definition is elegant, it suffers from practical drawbacks: computing the full eigendecomposition is computationally expensive, even prohibitive for large graphs, with a complexity of $O(|\mathcal{V}|^3)$. Moreover, the learned filters often lack spatial localization, making them difficult to transfer between different graphs.

To address the computational and generalization challenges of spectral graph convolution, researchers developed polynomial approximations of the spectral filter, such as Chebyshev polynomials [Hammond et al., 2011]. Instead of explicitly computing the eigenvectors **U** of the graph Laplacian and applying a filter in the spectral domain, these polynomial methods express the convolution as a series of powers of the Laplacian applied directly to the node features. This formulation effectively eliminates the need to compute **U** and allows the convolution to be implemented as a sequence of local, neighborhood-based operations—essentially aggregating information from each node's immediate or multi-hop neighbors.

This insight naturally bridges the gap between spectral methods and spatial methods and paved the way for the development of more scalable models. In particular, Kipf and Welling [2017] proposed the GCN, which simplifies the polynomial filter to a first-order neighborhood aggregation step with learnable weights (Equation 2.29).

$$\mathbf{H}^{(l+1)} = \sigma\left(\mathbf{D}^{-1/2}\tilde{\mathbf{A}}\mathbf{D}^{-1/2}\mathbf{H}^{(l)}\mathbf{W}^{(l+1)}\right), \quad l = 0, \ldots, L \tag{2.29}$$

In this equation, $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix with added self-loops and **D** is the corresponding degree matrix. The steps involved in this operation can be summarized as (1) aggregating features from the immediate neighbors of each node, followed by (2) a linear transformation and (3) a nonlinearity, providing a clean and familiar way of approximating spectral convolution in the spatial domain with a new form of neural networks that respect graph symmetries.

It may not even be strictly necessary to include nonlinearities at each layer to implement effective spatial graph convolution. Wu et al. [2019] demonstrated that by collapsing multiple layers of weight matrices and removing intermediate nonlinearities, the Simple Graph Convolution (SGC) can be viewed as a polynomial filter over the graph Laplacian. This formulation effectively approximates the behavior of GCNs of variable depths through repeated neighborhood aggregation steps, while also highlighting the continuum between spectral methods and their polynomial approximations in the spatial domain.

By replacing the spectral filter $g(\Lambda)$ with trainable parameters in the spatial domain, graph convolutions can be learned end-to-end as efficient, scalable, and transferable operations on large graphs. This reformulation shifts the focus from the spectral domain to a spatial perspective, where convolution corresponds to *message-passing over node neighborhoods*—a key idea underlying modern GNNs.

3.2 From Graph Convolution to Message-Passing Schemes

The previous subsection provided the theoretical backbone that takes us from the concept of graph convolution to its implementation through message-passing in the spatial domain. As such, message-passing schemes are the core of modern GNNs, where nodes iteratively update their representations by aggregating information from their neighbors. This process enables the network to capture dependencies and patterns within the graph structure.

Bronstein et al. [2021] categorize message-passing architectures into three primary paradigms: convolutional, attentional, and full message passing. These paradigms are visually represented in Figure 2.3 and conceptually described in the following paragraphs.

Convolutional Convolutional methods aggregate information from neighboring nodes using fixed, typically uniform, weights. The updated representation of each node is computed by an average/sum/max operation on the features of its neighbors (possibly including itself), followed by a learned transformation. This approach treats neighbor contributions equally (or with fixed weights) and forms the backbone of models like the GCNs [Kipf and Welling, 2017] and the ChebNet [Defferrard et al., 2016]. Other notable convolutional GNNs include GraphSAGE [Hamilton et al., 2017], which extends the aggregation scheme to mean, max, or LSTM-based pooling; GIN [Xu et al., 2019], which uses a sum aggregator and an MLP for feature transformation; GCNII [Chen et al., 2020], which adds residual connections; and GPR-GNN [Chien et al., 2020], which incorporates personalized PageRank-inspired propagation.

Formally, the GCN update for a node u at layer $l + 1$ is expressed by Equation 2.30, where $\mathbf{h}_u^{(l+1)} \in \mathbb{R}^d$ is the feature vector of node u at layer l, $\mathcal{N}_u$ denotes its neighbors, $\deg(u)$ is the degree of node u, $\mathbf{W}^{(l+1)}$ is a learnable weight matrix, and σ is a nonlinear activation function. This formal definition is analogous to the matrix form definition of GCNs in Equation 2.29.

$$\mathbf{h}_u^{(l+1)} = \sigma\left(\sum_{v\in\mathcal{N}_u\cup\{u\}} \frac{1}{\sqrt{\deg(u)}\sqrt{\deg(v)}}\mathbf{W}^{(l+1)}\mathbf{h}_v^{(l)}\right) \tag{2.30}$$

Attentional Attentional methods extend this idea by learning to assign different importances (or weights) to different neighbors. These models compute attention coefficients that quantify the relative contribution of the features of each neighbor to update the target node. GAT [Veličković et al., 2018] is a prime example of this paradigm, where attention coefficients are learned during training and allow the network to focus on the most relevant neighbors dynamically. Several variants of attentional GNNs have emerged beyond the original GAT, including GATv2 [Brody et al., 2022], which enhances expressivity with a more flexible attention mechanism, and AGNN [Thekumparampil et al., 2018], which leverages a cosine similarity-based attention. These models share the common principle of learning dynamic, data-driven attention coefficients that weight neighbor contributions adaptively.

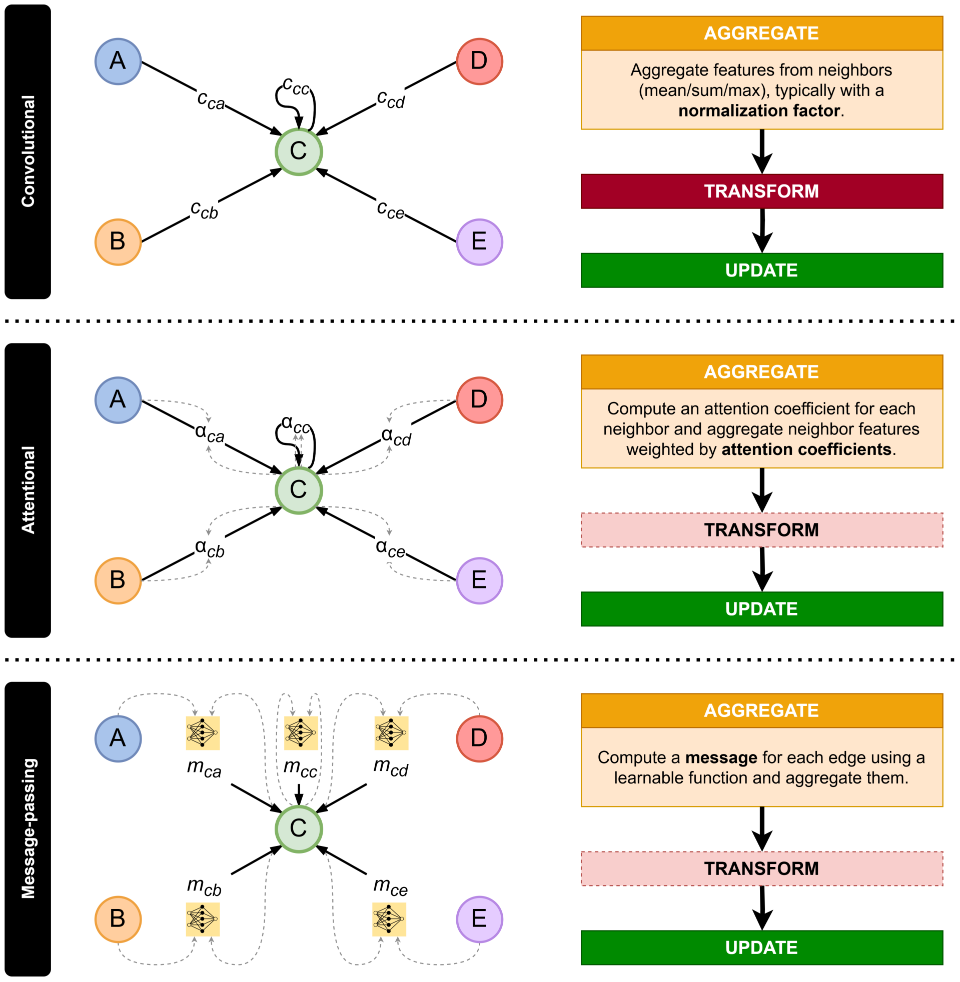

Convolutional
A
B
C
D
E
c_{ca}
c_{cc}
c_{cd}
c_{cb}
c_{ce}
AGGREGATE
Aggregate features from neighbors (mean/sum/max), typically with a **normalization factor**.
TRANSFORM
UPDATE
Attentional
α_{ca}
α_{cc}
α_{cd}
α_{cb}
α_{ce}
AGGREGATE
Compute an attention coefficient for each neighbor and aggregate neighbor features weighted by **attention coefficients**.
TRANSFORM
UPDATE
Message-passing
m_{ca}
m_{cc}
m_{cd}
m_{cb}
m_{ce}
AGGREGATE
Compute a **message** for each edge using a learnable function and aggregate them.
TRANSFORM
UPDATE

Figure 2.3: Convolutional, attentional and message-passing paradigms for GNN layers. [Left] Data flow visualization (inspired by Bronstein et al. [2021]). [Right] Computational flow description: *aggregate*, how neighbor information is gathered; *transform*, linear transformation of the aggregated features with learnable weights; *update*, combine the transformed message with the node's existing state (may include non-linearity, residual connections or gating). Dashed boxes correspond to optional steps.

Equation 2.31 shows how the embedding of node u at layer $l+1$ is updated in GAT, with the attention coefficients $\alpha_{uv}^{(l+1)}$ computed by Equation 2.32. Here, $\mathbf{a}^{(l+1)}$ is a learnable attention vector, $\|$ denotes concatenation, and LeakyReLU is a nonlinear activation function.

$$\mathbf{h}_u^{(l+1)} = \sigma\left(\sum_{v\in\mathcal{N}_u\cup\{u\}} \alpha_{uv}^{(l+1)}\mathbf{W}^{(l+1)}\mathbf{h}_v^{(l)}\right) \tag{2.31}$$

$$\alpha_{uv}^{(l+1)} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^{(l+1)\top}\left[\mathbf{W}^{(l+1)}\mathbf{h}_u^{(l)} \,\|\, \mathbf{W}^{(l+1)}\mathbf{h}_v^{(l)}\right]\right)\right)}{\sum_{j\in\mathcal{N}_u\cup\{u\}}\exp\left(\text{LeakyReLU}\left(\mathbf{a}^{(l+1)\top}\left[\mathbf{W}^{(l+1)}\mathbf{h}_u^{(l)} \,\|\, \mathbf{W}^{(l+1)}\mathbf{h}_j^{(l)}\right]\right)\right)} \tag{2.32}$$

Message-passing Full message passing generalizes both previous paradigms by allowing the message from each neighbor to be an arbitrary function of both the source and target node features, as well as potentially the edge features. In this setting, a learned function—often an MLP—computes the message on each edge. Models like the Message Passing Neural Network (MPNN) [Gilmer et al., 2017] and Graph Networks [Battaglia et al., 2018] exemplify this approach. Other methods in this category include GGNN [Li et al., 2016], which use gated recurrent units to update node states, and EdgeConv [Wang et al., 2019c], which computes edge-specific messages that are aggregated at each node. This paradigm provides maximal flexibility in modeling complex interactions, as each edge can contribute a distinct, learned message vector that is aggregated at the receiving node.

The formal update equations for node u at layer $l + 1$ can be split into message computation (Equation 2.33) and the actual update function (Equation 2.34). Here, $M^{(l+1)}$ and $U^{(l+1)}$ are learnable message and update functions, respectively, and $\mathbf{e}_{uv}$ denotes the features of the edge between nodes u and v.

$$\mathbf{m}_u^{(l+1)} = \sum_{v\in\mathcal{N}_u} M^{(l+1)}\left(\mathbf{h}_u^{(l)}, \mathbf{h}_v^{(l)}, \mathbf{e}_{uv}\right) \tag{2.33}$$

$$\mathbf{h}_u^{(l+1)} = U^{(l+1)}\left(\mathbf{h}_u^{(l)}, \mathbf{m}_u^{(l+1)}\right) \tag{2.34}$$

These paradigms reflect a trade-off between computational efficiency, expressiveness, and interpretability. Convolutional methods are efficient but less flexible; attentional methods introduce adaptive weighting; and full message passing enables the richest modeling at the cost of higher computational complexity. Together, they form the core intra-layer design configurations for modern GNN architectures.

Remark: Transformer-Based Graph Models

Besides GNNs, *transformer-based graph models*, such as Graphormer [Ying et al., 2021] and GTN [Yun et al., 2019], have emerged as powerful alternatives for graph representation learning. These architectures *compute attention scores globally across all node pairs*, incorporating graph structure indirectly through various encodings (e.g., shortest-path distances, Laplacian eigenvectors).

While GNNs only aggregate messages from structurally adjacent nodes, transformer-based models bypass the notion of local neighborhood aggregation altogether. Instead, they let the model itself decide which interactions are most important via learned attention weights—typically between any pair of nodes—, potentially ignoring immediate adjacency relationships entirely. While both types of models adaptively weight node interactions, *this key difference in how graph structure is integrated fundamentally changes how information is propagated*, with transformer-based models trading strict structural constraints for greater modeling flexibility and the ability to capture long-range dependencies and different relational patterns. This global attention approach can come at the expense of higher computational complexity, increased memory usage, and potentially less scalability on large graphs, making it crucial to consider the specific requirements of the task and dataset when choosing between local and global architectures.

Given their fundamentally different approach and decoupling from local neighborhood-based message-passing, these models will not be considered further in our analysis, despite their notable empirical successes.

3.3 Architectural Design Choices

Building upon the foundational message-passing paradigms, the architectural design of GNNs encompasses a broad and structured space. You et al. [2020] introduced a comprehensive framework categorizing GNN design into three principal dimensions: intra-layer design, inter-layer design, and learning configuration (depicted in red, orange and green in Figure 2.4, respectively). This taxonomy facilitates systematic exploration of the diversity of GNN architectural design options. The next paragraphs elaborate on them.

Intra-layer Design This dimension focuses on the operations within a single GNN layer, which ultimately define the core layer type and thus give rise to new GNNs. Key components include the choice of *aggregation functions* (e.g., sum, mean, max) that determine how information from neighboring nodes is combined, *activation functions* (e.g., ReLU, sigmoid) that introduce non-linearities, *normalization techniques* (e.g., batch normalization) that improve stability and convergence, and the incorporation of

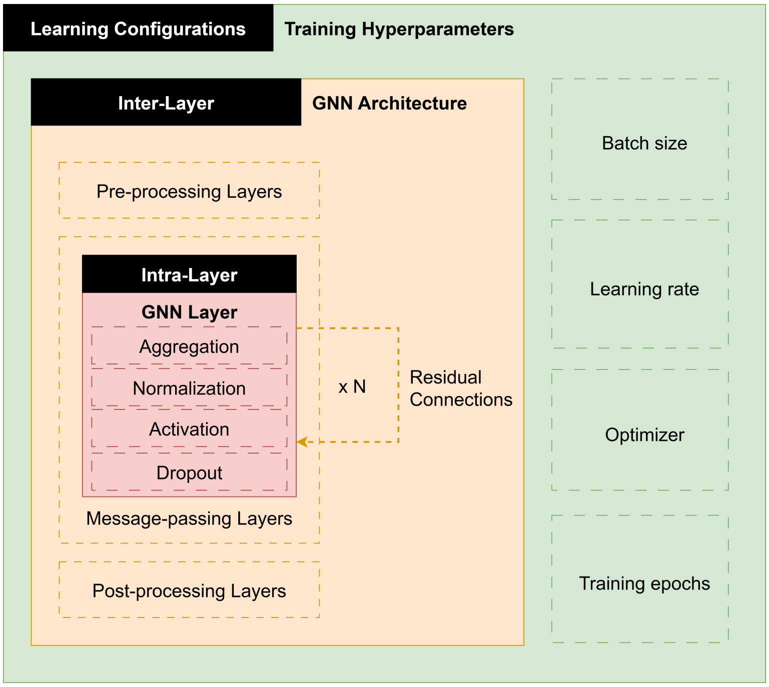


Figure 2.4: Categorization of GNN design dimensions, summarizing the framework proposed by You et al. [2020]. The figure illustrates the key components and choices involved in GNN architecture design according to their taxonomy.

dropout to mitigate overfitting by randomly deactivating a subset of neurons during training. For instance, GIN [Xu et al., 2019] emphasizes sum aggregation to achieve maximum expressive power, while GraphSAGE [Hamilton et al., 2017] introduces neighborhood sampling with various aggregation strategies, including mean and LSTM-based aggregators.

Importantly, while these methods represent different layer types, it is useful to recognize that they all operate within a shared underlying design space. By treating these intra-layer design dimensions as modular choices, researchers can systematically explore new architectures by flexibly recombining these dimensions, effectively traversing the landscape of existing models and uncovering novel configurations.

Inter-layer Design Inter-layer design encompasses the strategies for connecting and combining the outputs of different GNN layers. This includes connectivity design, using *residual connections* similar to those initially proposed by He et al. [2016] or in adapted forms, which add outputs from earlier layers to later ones to mitigate vanishing gradients and enable the training of deeper architectures. For example, GCNII [Chen et al., 2020] employs initial residual connections and identity mapping to facilitate deeper models.

Beyond connectivity, You et al. [2020] further decompose inter-layer design into additional sub-dimensions: *pre-processing layers* (e.g., initial node feature transformations or embeddings), *message-passing layers* (the core GNN layers themselves), and *post-processing layers* (e.g., final classification or readout functions). Each sub-dimension offers different options for controlling how information flows and interacts between layers, enabling models to create several levels of local and global representations. Collectively, these design choices shape the model's capacity to learn complex, hierarchical patterns across the graph, making inter-layer design a critical aspect of GNN architecture.

Learning Configuration Learning configurations refer to hyperparameter choices and training strategies that impact model performance, such as learning rate, weight decay, batch size, and optimizer selection. These aspects are common across deep learning architectures and share many of the same challenges as hyperparameter tuning in standard neural networks (e.g., CNNs and RNNs). The key distinction in GNNs is that these choices must be integrated with the unique graph structure and message-passing dynamics, which can affect convergence and generalization. For example, in standard deep learning, batch size typically refers to separate, independent samples, but in GNNs, batching often means sampling mini-batches of nodes from a larger graph. This requires techniques such as neighborhood sampling (e.g., GraphSAGE [Hamilton et al., 2017]) or cluster-based partitioning (Cluster-GCN [Chiang et al., 2019]) to manage memory and computation while preserving neighborhood information during training. Hence, while learning configurations may appear familiar, they often require adaptation to the nuances of GNN architectures and the target graph data, highlighting the interplay between algorithm design and data structure in this domain.

While the structured design space of GNNs aids in systematically exploring architectural choices, it can quickly become intractable due to the combinatorial explosion of possible configurations. For instance, You et al. (2020) identified over 315,000 distinct GNN designs under their framework, underscoring the vastness of this space. This complexity often surpasses that encountered in other deep learning domains, primarily because GNNs must account for both model architecture and the intricacies of graph structures [Zhou et al., 2020].

To navigate this expansive space, various automated search methods have been employed, including Neural Architecture Search (NAS) [Gao et al., 2020; Zhou et al., 2022], random search, and Bayesian optimization. While these techniques can identify high-performing architectures, they frequently lack interpretability and fail to provide theoretical insights into why certain configurations excel. Moreover, they often do not consider the specific characteristics of the graph data or the task at hand, limiting their applicability across diverse scenarios. This underscores the pressing need for more explainable and adaptive search strategies that explicitly consider the diverse and complex nature of graph data—ranging from varying sizes and structures to multiple levels of feature information—and that can align GNN architectures more closely with the unique characteristics and learning objectives of each task.

3.4 Learning Challenges

GNNs have shown remarkable performance in a wide range of tasks, but their learning process is often challenged by several key issues: oversmoothing, oversquashing, and harmful heterophily. These challenges are not isolated; they are often entangled with one another and deeply influenced by the underlying graph structure, making them complex to diagnose and mitigate.

Oversmoothing Oversmoothing in GNNs occurs when, as more layers are stacked, node representations become increasingly similar, ultimately converging to nearly constant embeddings that hinder class separation. This is a fundamental limitation in deep GNNs, where repeated neighborhood aggregation tends to homogenize node features across the graph [Rusch et al., 2023a]. A defining characteristic of this phenomenon is the collapse of node embeddings toward indistinguishability—regardless of class membership—which severely degrades the model's ability to learn meaningful decision boundaries.

Various metrics have been proposed to quantify oversmoothing, with Dirichlet energy being one of the most commonly used due to its ability to capture the smoothness of node features over the graph structure [Oono and Suzuki, 2019; Jaiswal et al., 2022]. However, recent work has highlighted key limitations of this measure. In some cases, Dirichlet energy can remain non-zero even when node embeddings have effectively collapsed, potentially obscuring the onset of oversmoothing [Rusch et al., 2023a; Zhang et al., 2025].

For illustrative purposes, Figure 2.5 shows a more intuitive lens that involves the classification performance degradation to the point of random guess and the direct inspection of the variance of node embeddings (at the last layer) as a function of propagation depth. As depth increases, accuracy initially holds but then sharply declines toward chance level, marking the loss of discriminative capacity. In parallel, as we approach oversmoothing, the variance across node embeddings shrinks toward zero, providing concrete evidence of representational collapse. The co-occurrence of these trends highlights how oversmoothing manifests both in the model's outputs and in

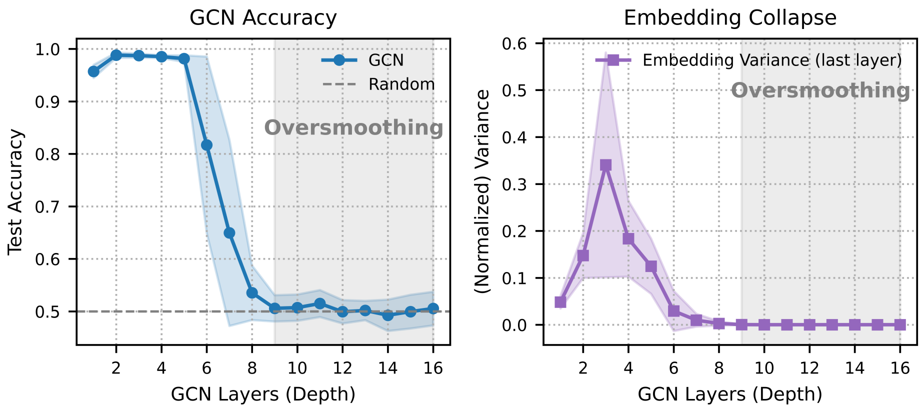


Figure 2.5: Empirical illustration of the oversmoothing effect. As depth increases, classification accuracy declines toward chance level [left], while node embeddings collapse to nearly identical representations, evidenced by a sharp drop in embedding variance to near zero values [right]. This reflects the core dynamic of oversmoothing: the progressive loss of representational diversity with depth.

its internal representations, offering a clear, empirical manifestation of oversmoothing that aligns with its conceptual definition—the loss of representational diversity across the graph—, evidencing that this phenomenon is not merely a theoretical issue but a measurable and observable failure mode in deep graph models.

Oversquashing Oversquashing refers to the compression of information from exponentially many distant nodes into fixed-size embeddings through a limited number of aggregation steps. This bottleneck hampers the ability of GNNs to capture long-range dependencies effectively [Alon and Yahav, 2021]. One way to analyze this phenomenon is through the concept of effective resistance, which measures how "easy" it is for information to travel between nodes in the graph [Topping et al., 2022]. By modifying edges to reduce effective resistance, recent methods have demonstrated improvements in mitigating oversquashing [Black et al., 2023].

Harmful Heterophily Heterophily arises when connected nodes are dissimilar, contradicting the assumption in many GNN architectures that neighboring nodes share similar features or labels. This mismatch can degrade performance as it complicates the aggregation of meaningful information. However, this problem can be much more intricate than that. Simply quantifying heterophily fails to capture nuanced aspects of node interactions, leading to controversies about whether heterophily is always detrimental. Recent research shows that heterophily may not be harmful at all, depending on the graph structure, attributes and learning objective [Luan et al., 2022; Arnaiz-Rodriguez and Errica, 2025]. These findings are a proof of how limited is the current understanding of how graph attributes and topology effectively impact GNN performance, which is a strong motivation for more dedicated analyses on this interplay and will be revisited in Chapter 3.

Interplay and Entanglement These learning challenges are deeply interconnected and the lack of a proper diagnosis of what is affecting a certain network architecture can hinder the identification of learning bottlenecks. The consequences of simply attributing degradation of performance to one of these causes without fully diagnosing and detangling the underlying problem have been pointed out in very recent works [Arnaiz-Rodriguez and Errica, 2025]; however, this view is not yet fully widespread in the GNN research community. For example, oversmoothing can be a cause for performance degradation; but it is not necessarily what makes the embeddings generated by a certain architecture less distinguishable—the architecture might simply not be appropriate for the graph learning task. Bottlenecks of information are typically attributed to oversquashing, but those bottlenecks can be computational or topological, which are two very different sources that require different approaches to handle. And the dichotomy homophily-heterophily alone has also been shown not to explain different learning patterns across GNN architectures. All of these learning challenges are rooted in or related to the topology of the graph, its attributes and the nature of the task itself. This entanglement means that solutions addressing one issue may inadvertently influence another, highlighting the need for holistic and adaptive strategies that consider the graph's structure and task-specific characteristics.

Addressing these challenges is crucial for advancing GNNs in real-world applications. While various mitigation techniques have been proposed, a deeper theoretical understanding and more reliable metrics are still needed to diagnose and alleviate these issues effectively, starting in an adequate diagnosis of the learning bottlenecks through dedicated analyses.

4 Discussion

Through a critical examination of the contents of this chapter, it becomes clear that there is an intricate relationship between graph geometry, learning possibilities and the exploration of the GNN design space. This section synthesizes these insights, emphasizing the challenges and opportunities that arise when designing effective GNN architectures. By highlighting key theoretical and practical considerations, this discussion aims to provide a cohesive understanding of the complexities inherent to graph-based deep learning.

The Curse of Dimensionality and Beyond

An important take-away message from this chapter is that GNNs operate in a fundamentally different data regime than traditional deep learning models designed for Euclidean data. Graphs exhibit highly flexible dimensions: they may contain an arbitrary number of nodes and edges, with a topology that is not fixed but varies across samples and tasks. This flexibility exacerbates the classic "curse of dimensionality" [Bellman and Kalaba, 1959], which describes how the volume of the input space grows exponentially with its dimensionality, rendering learning increasingly difficult. In the case of graphs, the inductive biases embedded in geometric deep learning are essential to mitigate this curse by enforcing structure in otherwise vast and sparse relational spaces [Bronstein et al., 2021].

Message-passing frameworks—central to most modern GNNs—implicitly incorporate these geometric inductive biases by enforcing permutation invariance and locality, thus regularizing the learning process [Battaglia et al., 2018; Gilmer et al., 2017]. However, despite these biases, the high flexibility in node connectivity, edge types, and attribute distributions continues to challenge the generalizability and robustness of GNNs [Dwivedi and Bresson, 2021]. Moreover, when moving beyond simple homophilic graphs (where connected nodes share similar attributes) to certain heterophilic or multi-relational graphs, these challenges intensify, as local neighborhoods can become highly noisy or uninformative [Yan et al., 2022]. This underscores the need for design choices that not only incorporate geometric priors but are also informed by the diverse topologies and attribute structures of real-world graphs.

Searching the Design Space of GNNs

Beyond the foundational geometric biases, the design space of GNNs is vast and highly combinatorial [You et al., 2020]. Each layer's aggregation function, update mechanism, attention mechanism, residual connections, and normalization choices can lead to significantly different model architectures; yet, the practical gains of exploring such a vast space of configurations remain ill-defined. This combinatorial explosion can quickly render manual exploration intractable, especially as graphs themselves vary in size and connectivity. Unlike images or sequences, graphs lack a regular grid or consistent sequential structure, complicating the evaluation and comparison of architectures across datasets and tasks.

Furthermore, intra-layer and inter-layer design choices often interact with the underlying graph structure in non-trivial ways. For instance, residual connections may mitigate oversmoothing, but this does not guarantee that such configuration will lead to more discriminative representations, as we will see in the next chapters. Similarly, attention mechanisms have the potential to help distinguish important edges but also introduce additional complexity in capturing relevant dependencies, as attention coefficients remain controversial as interpretable measures of edge connectivity importance [Jain and Wallace, 2019].

Despite advances in automated architecture search [Gao et al., 2020; Zhou et al., 2022], no single architecture configuration has emerged as universally superior across benchmarks and tasks. This highlights a key limitation of current search methods: they often optimize performance in a purely empirical yet task-agnostic way, without explicitly connecting architecture choices to the properties of the graphs on which they are applied. Thus, while geometric biases provide a starting point, further progress requires systematic integration of task-specific and graph-specific priors into the definition of relevant search spaces.

Connection with Graph Properties and Attributes

This discussion points us to the fact that GNN design is tightly coupled with the underlying properties of the target graph. Since no single architecture consistently outperforms others across benchmarks, it becomes essential to ask: *what are the key design choices for a given graph task?* One promising direction is to develop more principled frameworks that align graph attributes and properties with architectural design choices. For example, some graph properties can significantly impact the effectiveness of GNNs of different configurations, much more than the architectural tweaks that distinguish their intra-layer setups.

To illustrate this point, Table 2.2 presents a small-scale experiment comparing GNN performance across graphs with varying density and homophily, but fixed feature information (details on the experimental setup can be found in Appendix A.2). In this analysis, each configuration was evaluated over ten independent runs using an 80-10-10% train/validation/test split; the resulting accuracies were grouped either by layer type or by graph property setting and compared for statistical significance via paired t-tests (Table 2.2, columns for layer type comparisons and rows for graph property comparisons). Interestingly, while graph properties (density and homophily) substantially influence classification performance, the choice of layer type (GCN, GAT, GIN, or SAGE) often yields comparable results within a given property configuration. For instance, all layer types achieve perfect accuracy in high-density, high-homophily graphs, while performance drops uniformly across all layers in medium-density, medium-homophily graphs. This suggests that, in some cases, graph properties can

Table 2.2: Performance of various GNN layer types (GCN, GAT, GIN, SAGE) across different graph densities and homophily levels. Results demonstrate that while layer type often yields similar performance, underlying graph properties can significantly influence classification outcomes.

Properties		**Layer Type**					
Density	Homophily	GCN	GAT	GIN	SAGE		
Low	Low	0.85	0.84	0.78	0.85	←	
Low	Medium	0.88	0.88	0.88	0.88	←	Found statistically
Low	High	0.78	0.78	0.78	0.77	←	significant differences
Medium	Low	0.79	0.79	0.79	0.79	←	between most
Medium	Medium	0.71	0.71	0.71	0.71	←	combinations
Medium	High	0.81	0.80	0.77	0.81	←	$\therefore$ **Graph properties**
High	Low	0.78	0.82	0.80	0.86	←	**influence classification**
High	Medium	0.72	0.73	0.72	0.72	←	**performance**
High	High	1.00	1.00	1.00	1.00	←	
		↑	↑	↑	↑		

No statistically significant differences
between layer types

be more decisive than intra-layer configurations—highlighting the potential benefits of integrating graph analysis into model design, potentially avoiding an expensive combinatorial search over near-equivalent intra-layer setups and focusing more on other configurations. This approach can influence the creation of more relevant, narrow search spaces, informed by the properties and attributes of the underlying graph.

This insight has profound implications. Instead of focusing solely on optimizing architectures through black-box search, future research can explore frameworks that explicitly connect graph attributes to architectural design decisions. Such approaches would not only enhance interpretability but also ensure that GNN models are better tailored to the nuances of their target domains.

In sum, the challenges facing GNNs—rooted in the curse of dimensionality, the vast combinatorial design space, and the complex interplay of graph properties—underscore the need for a more systematic approach to architecture design. Geometric inductive biases remain crucial, but they are not sufficient on their own. Progress requires embracing the heterogeneity of graphs, developing design search strategies informed by graph characteristics, and creating frameworks that align architecture with task-specific relational patterns. This new empirical paradigm can promote a holistic integration of GNN dimensions, enabling them to reach their full potential in modeling the rich relational complexity of real-world data.

5 Conclusion

The convergence of graph learning and deep learning holds the promise of unlocking insights from complex, relational data. Throughout this chapter, the multifaceted relationship between theoretical underpinnings and practical implementation has been examined. The curse of dimensionality emerges as a fundamental challenge, especially within the flexible and high-dimensional landscape of graphs, underscoring the critical role of geometric inductive biases. Simultaneously, the exploration of the GNN design space highlights the absence of a universally optimal architecture, revealing a need for principled strategies that align model design with task-specific graph characteristics. The empirical evidence presented affirms that while significantly different layer configurations can perform similarly, underlying graph properties such as density and homophily can profoundly affect outcomes, illuminating the path toward more targeted model development. As the field progresses, bridging the theoretical intricacies of graph structures with the practical demands of deep learning stands as a pivotal endeavor, ensuring that advancements in AI are not only mathematically sound but also impactful across real-world scenarios.

3 Feature-Structure Interplay in Graph Neural Networks

The previous chapters have elaborated on the historical, theoretical and practical background that made GNNs emerge as the default approach for graph representation learning in machine learning tasks. These models have clearly demonstrated their ability to learn from relational data by achieving multiple successful applications in diverse domains such as drug design [Wong et al., 2024], biology [Jha et al., 2022] and complex physics simulations [Sanchez-Gonzalez et al., 2020]. Despite this fact, not everything is known about what is essential for these networks to learn and, crucially, to excel. In particular, one can easily find examples in which *some GNNs are unable to learn useful node/graph representations and underperform when compared to structure-agnostic counterparts*, such as MLPs.

In typical semi-supervised node classification problems, GNNs make class predictions using two sources of information: the individual node properties described by the feature vectors associated with each node, and structural information represented by the relationships (edges) between the different nodes. The main idea behind using graph representations is that relational information can provide additional information to solve the target task, e.g. friendship links in a social network can be predictive of the interests of a person. Given that we are augmenting the node properties with additional information from node relations, one might assume that GNNs always outperform feature-only methods that do not exploit any structural information. However, recent benchmark results show some state-of-the-art GNNs performing on-par or worse than basic MLPs on a variety of node classification tasks, as we will see in Section 1. These results hint at the fact that the use of structural information by GNNs is not always helpful and may even be detrimental to classification performance, not only for the simplest network architectures but also for the more recent and complex ones (Figure 3.1).

While several authors have investigated the phenomena of oversmoothing, heterophily and loss of expressivity in GNNs [Cai and Wang, 2020; Balcilar et al., 2021; Oono and Suzuki, 2019; Yan et al., 2022], these works frequently focus on representational power as related to the depth of the GNNs or to a limited analysis of graph structure centered in the concepts of homo-/hetero-phily and, more critically, consider feature and structure information as a whole. This study aims to decouple the influence of structural and feature-level information on graph learning to assess the extent to which the underlying assumptions of GNN architectures affect their performance. To achieve this, the problem of semi-supervised node classification is examined through a systematic and empirical investigation of the roles played by structural and feature information across various GNN methods.

Section 1 formalizes the problem and assumptions under analysis throughout this chapter. After that, the chapter is organized around two primary contributions. First, it introduces an *empirical, model-agnostic methodology for disentangling the respective influences of node features and graph structure on the performance of GNN-based node classification* (Section 2). Second, it presents a set of *novel empirical findings that elucidate how structural and feature information individually contribute to GNN performance*, offering insights that may inform future developments in graph learning methods. These findings include the observation that simple GCNs require both meaningful structural and feature information to learn effective node representations; in the absence of one, the models struggle to distinguish informative signals from noise. Nevertheless, GCNs can substantially outperform feature-only baselines when structural information is highly relevant, even if feature signals are relatively weak (Section 3). Furthermore, Section 4 demonstrates that the absolute performance gains of GNNs over feature-only models can be quantified, revealing that these gains are closely tied to the capacity that a certain model has to exploit structural information relative to the informativeness of the available features.

Finally, the chapter concludes with a discussion that synthesizes the presented insights, highlighting their potential implications for future research in graph representation learning (Section 5). The discussion underscores how a deeper understanding of the interplay between structural and feature information can inform more effective and principled approaches to GNN development. This includes *considerations for the design of neural network architectures aimed at optimizing the extraction of discriminative representations tailored to specific tasks.*

Remark

The main contents of this chapter were published under the following venues:

Gomes, D., Ruelens, F., Efthymiadis, K., Nowe, A., & Vrancx, P. (2022). When are graph neural networks better than structure-agnostic methods?. In *I Can't Believe It's Not Better Workshop: Understanding Deep Learning Through Empirical Falsification @ NeurIPS'22.*

Gomes, D., Nowe, A., & Vrancx, P. (2024). Understanding Feature/Structure Interplay in Graph Neural Networks. In *Proceedings of the Third Learning on Graphs Conference (LoG 2024).*

1 Problem Statement

Despite the increasing sophistication of GNN architectures, their performance across datasets and predictive tasks remains inconsistent and often unpredictable. These empirical disparities highlight a fundamental gap in our understanding of how and when GNNs succeed. At the heart of this issue lies the dual reliance of GNNs on both node features and graph structure—two intertwined components that are integrated via local aggregation mechanisms. While graph smoothing, a byproduct of such aggregation, is frequently assumed to be beneficial, we will demonstrate that this effect can vary significantly depending on the task and the graph's properties. In this context, and throughout the chapter, we refer to *task* as the specific predictive objective under consideration—not only the distinction between classification and regression problems, but even alternative classification targets defined on the same graph—since each such objective induces its own distribution of labels, degree of feature informativeness, and relevance of the underlying topology.

This section articulates the central problem addressed in this chapter: the lack of a principled understanding of the interplay between feature and structural contributions in GNNs. We begin by examining the empirical inconsistencies observed across models, then discuss the implications of smoothing as both a strength and a limitation, and finally introduce our framework for reassessing the assumed benefits of smoothing through a controlled decoupling of feature and structure influences, which we shall adopt throughout the analyses of this chapter.

1.1 The Inconsistent Trends of GNN Performance

To understand the problem that motivated the present study, let us take a closer look at the performance trends of current state-of-the-art GNNs. Figure 3.1 combines the

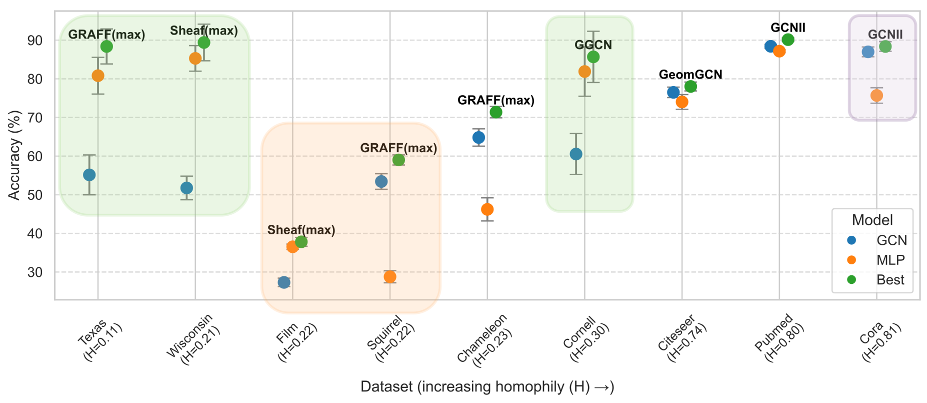


Figure 3.1: Performance comparison of GCN, MLP, and the best-performing model on nine benchmark datasets with varying levels of homophily (H) in ascending order along the x-axis. Results were extracted from the benchmark tables of Yan et al. [2022] and Giovanni et al. [2023]. Mean classification accuracy (± standard deviation) is reported for each model, and the best-performing method per dataset is annotated above their respective points. [Annotated Examples] *Orange*: homophily levels do not justify the observed disparity in performance; *Green*: structure-agnostic methods performing on-par with best performing GNN; *Purple*: simple GCNs performing on-par with best performing GNNs.

results of recent benchmark studies across datasets with varying degrees of homophily—a property that has been deemed crucial for the successful application of GNNs to solve a graph problem, as discussed in the previous chapter, leading to a body of literature inspecting its implications [Ma et al., 2022; Luan et al., 2023; Zhu et al., 2020]. Although Figure 3.1 presents an extensive set of results, it also underscores the complexity and partial understanding that currently characterizes the field of GNN research. The lack of clear, consistent performance patterns across different architectures highlights the challenges involved in designing effective GNN models across the spectrum of graph problems. In particular, the following observations illustrate this point:

1. *Homophily levels do not justify the observed disparity in performance*: Despite similar (low) homophily scores ($H = 0.22$), *Film* and *Squirrel* exhibit markedly different performance patterns (*orange* example in Figure 3.1). In *Film*, GCN performs poorly (27%), while MLP nearly matches the best model (37% vs. 38%). In contrast, *Squirrel* sees strong performance from GCN (53%) and poor results from MLP (29%). These opposing trends in graphs of similar homophily highlight that this property

alone cannot explain model behavior, suggesting that other factors—such as feature informativeness or graph connectivity—play a critical role.

2. *Structure-agnostic methods perform on-par with best performing GNN on several occasions*: Interestingly, MLPs—which entirely ignore graph structure—achieve performance that is competitive with, and occasionally close to, the best-performing GNNs in some cases. In the *Texas*, *Wisconsin*, and *Cornell* datasets, MLPs reach 81%, 85%, and 82% accuracy, respectively, trailing the best models by only a few percentage points, negligible if error margins are considered (*green* example in Figure 3.1). These results demonstrate that in certain settings, feature information alone is sufficient to achieve high predictive performance, and that incorporating graph structure does not always yield a clear advantage.

3. *Simple GCNs perform on-par with best performing GNNs on some occasions*: In more homophilic settings, such as *Cora* ($H = 0.81$), classical GCNs remain highly competitive (*purple* example in Figure 3.1). This narrow margin implies that, in favorable yet ill-defined conditions, the performance ceiling of sophisticated architectures may be only marginally higher than that of simpler baselines, questioning the need for added complexity in such regimes.

4. *No single GNN architecture performs the best across the spectrum*: A striking pattern in Figure 3.1 is the lack of a universally dominant architecture. The best-performing models vary significantly across datasets: GRAFF on *Texas*, GGCN on *Cornell*, GeomGCN on *Citeseer*, and GCNII on *Cora* and *Pubmed*. This heterogeneity in winners underscores the absence of a one-size-fits-all solution in graph learning, and highlights the importance of architectural selection tailored to dataset-specific properties.

Such empirical ambiguities underscore the relevance of the research question of this thesis—what are the key requirements for the effective performance of GNNs—and the crucial need for a *systematic and principled investigation into the separate and combined influences of feature and structure components on model efficacy*. As such, a central challenge underlying the investigation presented in this chapter is the *disentanglement of feature-driven and structure-driven contributions to GNN behavior*.

GNNs, by design, integrate these two sources of information, feature information and graph structure, through joint aggregation mechanisms, rendering their effects difficult to isolate. However, a growing body of very recent research emphasizes the critical importance of decoupling feature and structural influences to achieve a more precise understanding of model performance, generalization capabilities, and failure modes [Castellana and Errica, 2023; Arnaiz-Rodriguez and Errica, 2025]. Without such decoupling, the advancement of interpretability, diagnostic tools, and the development of more robust graph learning architectures remains significantly constrained.

To enable such disentanglement, it is essential to first examine the mechanisms by which feature and structural signals interact within GNNs. One particularly influential mechanism is *graph smoothing*, which lies at the heart of many GNN operations. Understanding the dual role that smoothing plays—both as a potential enabler of signal enhancement and a risk factor for excessive smoothing—is critical for interpreting the outcomes of feature-structure interplay. This motivates a deeper look into how smoothing emerges from graph convolutions and under what conditions it can contribute positively or negatively to task performance.

1.2 The Dual Nature of Graph Smoothing

Beneficial Smoothing Effect Graph convolutions, at their core, rely on consecutive local averaging operations, making them inherently local smoothers. This smoothing effect has been widely acknowledged as a key strength of GCN layers, especially in homophilic graphs, where nodes with the same class label tend to form tightly connected clusters. In such settings, smoothing can amplify useful signals by reinforcing intra-class similarities and filtering out noise, resulting in improved downstream performance compared to feature-only baselines such as MLPs (see "Beneficial Smoothing" area in Figure 3.2). Indeed, it is often this very smoothing that accounts for the empirical success of GCNs in homophilic domains.

Harmful Smoothing Effect However, while smoothing can be highly effective in homophilic settings, its utility is far from universal. In graphs where nodes are frequently connected to others from different classes—commonly referred to as heterophilic—the same smoothing mechanism can become a liability in some cases (recall the *orange* example of Figure 3.1—under comparable levels of homophily, the effectiveness of simple graph convolution operations can yield substantial performance benefits, while in other cases they offer little to no gains). These suboptimal cases correspond to situations in which local averaging may obscure discriminative features, diluting class-specific signals and thereby impairing model performance (see "Excessive Smoothing" area in Figure 3.2—bottom case—in which beneficial smoothing is never experienced). The observation that there harmful cases of heterophily, but not all heterophilic graphs exhibit the same learning patterns, challenges the assumption that homophily is a necessary or sufficient condition for GNN success. Rather than dismissing smoothing in these contexts, it is more accurate to recognize that homophily represents only one among many possible structural patterns that may support effective graph-based learning (as we will see later in the chapter). However, our current understanding of how alternative topological configurations interact with feature propagation remains limited, highlighting a need for broader structural theories beyond homophily alone.

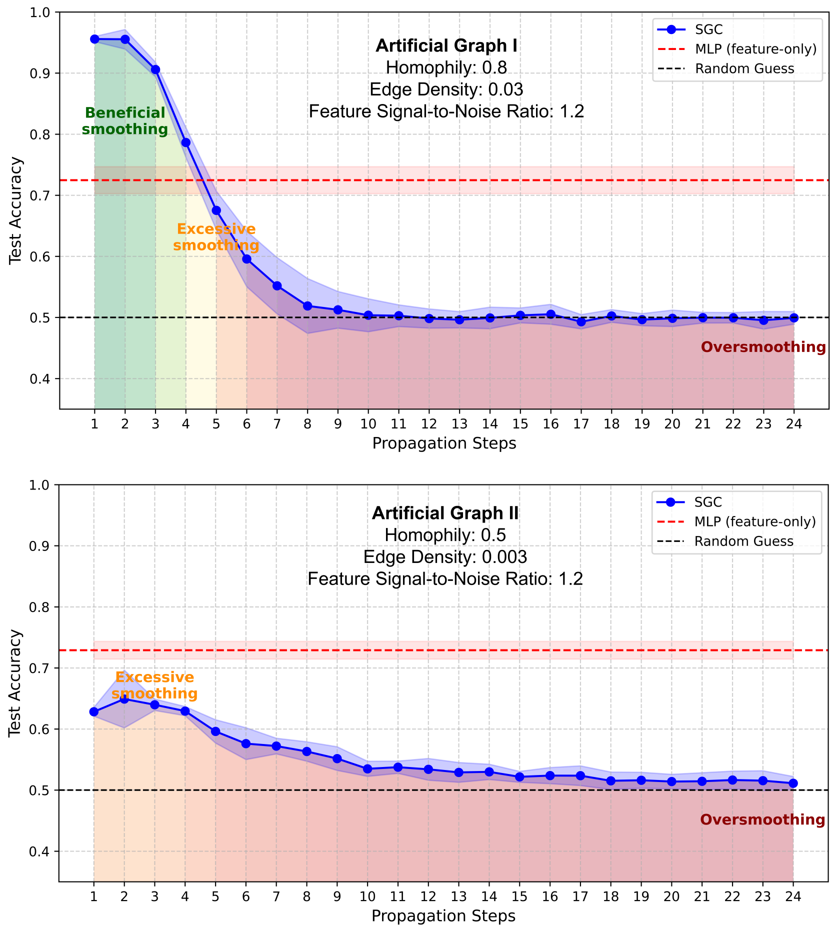


Figure 3.2: Effects of sequential graph convolutions on classification and representation quality. Using SGC [Wu et al., 2019] to control the number of propagation steps, we illustrate the impact of graph smoothing. [Top] In a homophilic synthetic graph, increasing propagation initially improves accuracy ("Beneficial smoothing"), then degrades it ("Excessive smoothing"), and eventually leads to near-random performance ("Oversmoothing"). [Bottom] In a graph with less informative structure, performance immediately drops below the MLP baseline, indicating no beneficial smoothing phase. Shaded regions indicate smoothing regimes.

Oversmoothing The potential drawbacks of smoothing become even more pronounced when multiple propagation steps are stacked. As aggregation is applied repeatedly, node representations tend to converge—a phenomenon known as oversmoothing (see "Oversmoothing" examples in Figure 3.2). This effect can severely limit the model's ability to maintain class-specific distinctions, especially in deeper architectures. As discussed in the previous chapter, oversmoothing leads to increasingly uniform node embeddings and ultimately erodes the discriminative power of the model. Together, these challenges emphasize that the effectiveness of graph smoothing is not inherent but context-dependent, shaped by subtle interactions between graph structure, feature space, and model depth.

This dual nature of graph smoothing—sometimes beneficial, sometimes harmful—challenges the notion of GCNs as a universally superior solution for graph-based learning. It has also driven the design of increasingly complex architectures that seek to balance or selectively apply structural aggregation. Yet, even among these advanced models, performance gains are often inconsistent across benchmarks, and statistically indistinguishable from simpler baselines when error bars are considered, as we saw in Figure 3.1. This raises a fundamental question about the conditions under which smoothing—via graph convolutions—truly enhances overall learning.

Rethinking Assumptions of Beneficial Smoothing in GNNs by Analyzing Feature/Structure Interplay

Building on the preceding context, this chapter begins by examining a central hypothesis: let us assume that *graph convolution operations—whether simple or advanced—consistently produce beneficial smoothing*. This assumption frequently serves as the implicit rationale for selecting a GNN architecture for a given graph learning task—namely, the expectation that local aggregation mechanisms will enhance class separability by effectively leveraging the structure of the graph. More explicitly, we can define this assumption as:

> *The local smoothing obtained through a given sequence of graph convolution operations leads to useful node embeddings, suitable for solving a downstream task.*

Throughout this chapter, we aim to critically examine the validity of this assumption by analyzing the decoupled influence of feature and structural information in GCNs and other GNN variants. Through a series of controlled experiments on both synthetic and real-world datasets, we evaluate simple and advanced models to uncover the nuanced interplay between smoothing, structural bias, and representational power. This investigation seeks to offer new insights into when and why smoothing aids or hinders learning, ultimately informing the development of more robust graph learning paradigms.

To achieve this, let us start by introducing our novel method for studying the separate influence of feature and structure components in GNN performance in the next section.

2 Feature-Structure Decoupling

In order to study when GNNs can and cannot learn useful node representations, we propose a new, model-agnostic method which intends to decouple the influence of feature and structure information in each node classification task and separately violate GNNs' assumptions of meaningful underlying structure and features. This method can provide a quantitative measure of how much each GNN is leveraging the structure vs. feature components, thus assisting the identification of learning bottlenecks.

2.1 Method

Graph mutations

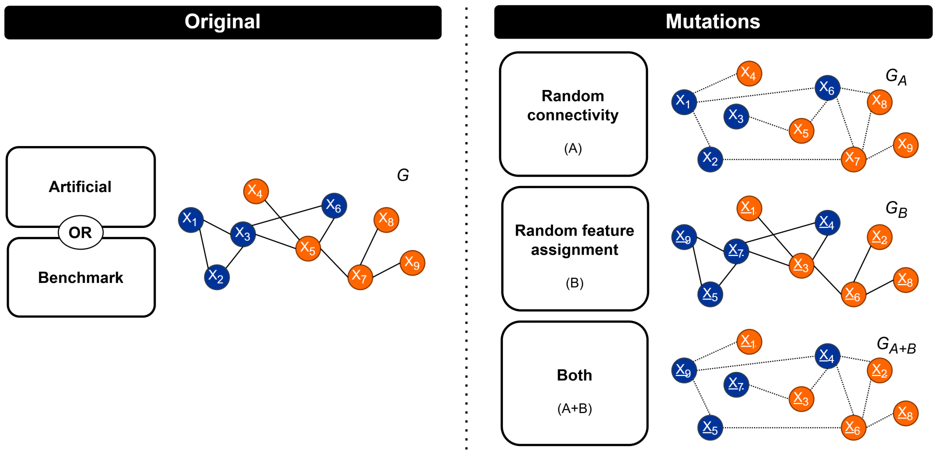


Figure 3.3: Graph transformations proposed in this work to methodically destroy structure and/or features while preserving the remaining graph attributes. For each original graph G, three combinations of feature/structure transformations are created (mutations).

We implement two operations designed to derive a set of mutations for each original graph. These operations are set to methodically destroy structure and/or feature information while preserving the remaining graph attributes (Figure 3.3). Each input graph (artificial or real-world; details on the generation of artificial graphs and on the benchmark datasets used in this study are provided in Appendix A.4) is submitted to transformations on the structural and feature levels:

- *Random connectivity*: shuffle columns of graph connectivity matrix;
- *Random feature assignment*: shuffle the attribution of feature vectors across all nodes, while keeping the original target label of each node.

These randomization steps ensure that, while some global statistics of the graph (e.g., number of nodes, overall edge density) remain unchanged, the relational or feature-based signals that could support the downstream predictive task are effectively disrupted. In other words, the randomized graphs retain the same basic size and number of connections as the originals but no longer encode the meaningful structural patterns or node-feature alignments that a GNN could exploit.

With these operations, we conceive up to three mutations (Figure 3.3) for each original graph whenever appropriate: a mutation with random connectivity but same feature distribution (G_A); another with same structural information but random feature assignment (G_B); and a final one with both transformations (G_{A+B}).

Comparison with baselines

Each original graph and respective mutations are then separately used to train two different types of models—a GNN, with no restrictions on complexity or architecture, and a structure-agnostic model (in this case, an MLP). These outputs can be interpreted on their own, i.e. to perform direct comparisons between mutations/original versions and draw conclusions based on performance trends, or yet be used to compute two metrics for measuring the separate contributions of structure (Equation 3.1) and feature (Equation 3.2) components, when appropriate. Finally, the comparison with the MLP evidences whether structure encoding using the GNN architecture under analysis is generally successful or harmful, i.e., GNN performance on the original graph is superior to feature-only MLP or not, respectively.

2.2 Measurement and Interpretation of Structure and Feature Contributions

To systematically assess the relative importance of structural and feature information in GNN performance, we introduce two metrics: *Structure Contribution* (C_{struct}) and *Feature*

Contribution (C_{feat}). These metrics aim to isolate and quantify the respective influence of each component on a model's predictive capability, using controlled graph perturbations.

Let us denote the predictive accuracy (or any relevant performance metric) of a GNN trained under different input configurations as follows:

- S_{orig}: performance on the original graph, containing both real node features and original connectivity;
- S_{randS}: performance when the graph structure is randomized, but original node features are preserved;
- S_{randF}: performance when the node features are randomized, but the original graph structure is preserved;
- S_{rand}: performance when both node features and graph structure are randomized.

These configurations allow us to measure performance degradation or improvement when one or both sources of information are corrupted. Based on this setup, we define the structure contribution by Equation 3.1 and feature contribution by Equation 3.2. Let us break them down.

Structure Contribution (C_{struct})

$$C_{\text{struct}} = (S_{\text{orig}} - S_{\text{randS}}) + (S_{\text{randF}} - S_{\text{rand}}) \tag{3.1}$$

Each term in this equation has a clear interpretation. The first term ($S_{\text{orig}} - S_{\text{randS}}$) captures the performance loss incurred by randomizing the structure while retaining the original features—i.e., the benefit of real structure in the presence of relevant features. The second term ($S_{\text{randF}} - S_{\text{rand}}$) measures the structural contribution when features are uninformative, isolating the added value of graph connectivity alone operating on uninformative feature content (by contrast with a fully randomized/uninformative graph).

Taken together, C_{struct} estimates *how much structural information contributes to the model's performance*, both when features are intact (i.e., how productive is the smoothing of the feature signal given the graph's structure) and when they are degraded (i.e., how structure alone can contribute to solve the graph problem), constituting a comprehensive metric that informs on the overall added-value of structure.

Feature Contribution (C_{feat})

$$C_{\text{feat}} = (S_{\text{orig}} - S_{\text{randF}}) + (S_{\text{randS}} - S_{\text{rand}}) \tag{3.2}$$

The terms in Equation 3.2 also have an analogous interpretation to those of the structure case. The first term ($S_{\text{orig}} - S_{\text{randF}}$) represents the performance drop due to feature randomization when structure is preserved, indicating the value of real features for the

GNN performance, while the second term ($S_{\text{randS}} - S_{\text{rand}}$) captures the isolated feature contribution when the structure is overall uninformative.

Thus, C_{feat} summarizes the benefit of node features in both informative and uninformative structure graph settings, ultimately *informing us about the added-value of the feature content towards solving the graph problem with a certain GNN.*

These metrics are designed to be structurally symmetric in formulation, meaning that the contributions of structure and features are assessed using parallel and balanced comparisons: each metric is computed by evaluating the performance impact of one component (structure or features) under both intact and randomized conditions of the other. This symmetry in design ensures interpretability and fairness, as it avoids methodological bias toward either modality while enabling meaningful side-by-side comparisons.

By incorporating both the gain from real information and the loss from its destruction, they offer a robust means of diagnosing what drives GNN performance for a given dataset and model. Moreover, they enable a fair comparison across GNN architectures and datasets, even when the underlying data distributions or topological properties differ substantially.

3 Decoupling Feature and Structure Influence in Controlled and Real-World Scenarios

Our investigation of the cases when GNNs can and cannot learn useful node representations is conducted by applying our previously introduced features/structure decoupling method to artificial graphs with certain engineered properties. In particular, we manipulate homophily and edge density (structural properties) and feature signal-to-noise ratio (SNR), due to their direct relation with the implicit assumptions of message-passing graph models.

This meticulous manipulation of properties enable us to experiment in a fully controlled scenario in which we can measure how GNNs respond to variations of the graph structure and the feature vector from an empirical perspective (Subsection 3.1). These experiments are then further extended by extrapolating our insights to more challenging settings by means of several real-world datasets, commonly used as benchmarks for machine learning on graphs, and more advanced network architectures (Subsection 3.2). All details concerning experimental setup, including artificial graph generation, benchmark datasets and model selection, training and evaluation can be found in Appendix A.4.

3.1 Graph Convolution on Artificial Graphs with Engineered Properties

While GCNs seem to be an efficient and straightforward method to apply machine learning to graphs, the fact that they perform on-par or worse than basic MLPs on certain tasks makes us question whether GCNs are able to encode graph structure productively for all tasks. *Is there always a practical benefit in encoding both feature and structure information for node representation learning with GCNs? And, if not, can we identify under which conditions is the exploitation of both levels of information detrimental for the classification task?* The following paragraphs provide an answer to this question by decoupling the influence of feature/structure in GCN's performance on artificial graphs with engineered properties using our new method (as introduced in Section 2).

To facilitate the interpretation of the controlled experiments, we briefly clarify the manipulated properties under analysis: *homophily* quantifies the tendency of nodes from the same class to be connected (recall Equation 2.13 of Chapter 2), while *edge density* reflects the overall proportion of observed vs. possible edges in the graph (Equation 2.12 of Chapter 2), thereby modulating the intensity of neighborhood aggregation. Feature SNR captures the relative separability of class distributions in feature space, defined as the ratio of inter-class variance to intra-class variance. This metric is bounded below by 1.0, which corresponds to the degenerate case where inter-class and intra-class variance are equal—i.e., class information is completely obscured by noise. Higher values indicate increasingly stronger alignment between feature vectors and class identity, with more informative and discriminative features.

It is also important to acknowledge that mutating the graphs inevitably alters some properties while leaving others intact. Randomizing connectivity typically disrupts node adjacency patterns and drives homophily toward roughly 0.5 in these balanced tasks. Likewise, shuffling feature vectors breaks feature-label alignment and lowers feature SNR toward its lower bound ($\approx$ 1.0, where inter- and intra-class variances converge). In contrast, global counts—number of nodes and total edges—are preserved, so edge density remains unchanged. Although this procedure does not allow fine-grained control over which secondary attributes are affected, it deliberately destroys the structural patterns and feature-label relationships that would otherwise guide GNN classification. This enables us to isolate how performance degrades when key sources of graph information are removed, while still being aware that mutations inevitably influence multiple attributes. However, for all analyses, the original graphs serve as the reference point, retaining their carefully engineered properties in which only one factor was varied while all others were held as constant as possible, which is critical for the analysis of performance drops that will follow. All reported table values correspond to the mean and standard deviation for the testset over 10 independent runs using a fixed 80-10-10% train/validation/test split, with further details on hyperparameter configuration and implementation provided in Appendix A.4.

Informative graph

Let us consider the simple case of binary node classification in a graph with both highly informative features (SNR = 1.5) and structure (homophily = 0.95; edge density = 0.06). Table 3.1 compares the performance of a 2-layer vanilla-GCN with that of a structure-agnostic model (MLP) on the original version of this graph and its mutations (uninformative versions).

Table 3.1: Node classification performance (mean accuracy ± standard deviation) of GCN and MLP models on an artificial graph *G* with highly informative structure and features, and the mutations of *G* with random connectivity (uninformative structure), random feature assignment (uninformative features), and both (uninformative structure and features).

		Structure			
		Informative		Uninformative	
Features	Informative	GCN	0.96 ± 0.02	GCN	0.69 ± 0.02
		MLP	0.92 ± 0.02	MLP	0.90 ± 0.04
	Uninformative	GCN	0.61 ± 0.04	GCN	0.48 ± 0.03
		MLP	0.50 ± 0.05	MLP	0.50 ± 0.03

While GCN exhibits adequate performance (superior to the MLP) when both structural and feature information are present, results show an evident drop when either is lost. Despite the fact that features are highly informative, GCN does not seem able to fully leverage them when the structure of the input graph is meaningless towards its inherent assumptions, leading to a significant loss of performance even relatively to its structure-agnostic counterpart. A similar drop is verified in the scenario where structure is preserved but feature information is lost, as GCNs are not able to aggregate neighborhood information in meaningful node representations, despite the highly informative structure of the input graph. We verify this behavior using shallow models of 2 message-passing layers, for which graph oversmoothing does not occur, as the appropriate performance in the informative scenario corroborates.

These results suggest that GCN models need both feature and structural information to be meaningful in order to learn useful node representations. When only one of these is present, the model does not seem to be able to separate the useful from meaningless information. This result makes sense given the intuition of GCN as a smoothing operator

Table 3.2: Node classification performance (mean accuracy ± standard deviation) of GCN and MLP (baseline) models on artificial graphs with fixed edge density (0.03) and features (SNR = 1.2) and different homophily (h). Results are shown for each original graph G and the respective mutations of G with random connectivity, random feature assignment, and both. Given its feature-only nature, a single MLP result is shown per feature transformation.

	h	Structure		MLP (baseline)
		Original	Random	
Original Features	0.2	0.78 ± 0.08	0.58 ± 0.03	0.71 ± 0.03
	0.5	0.60 ± 0.02	0.61 ± 0.03	
	0.8	0.81 ± 0.03	0.59 ± 0.03	
Random Feature Assignment	0.2	0.51 ± 0.05	0.48 ± 0.04	0.49± 0.03
	0.5	0.53 ± 0.04	0.50 ± 0.03	
	0.8	0.49 ± 0.03	0.49 ± 0.02	

[Li et al., 2018]. Blindly aggregating either features of dissimilar nodes due to lack of structural information or combining non-informative features of similar nodes does not extract useful node descriptions.

Varying Structural Properties

Given the empirical verification that GCN performance can be closely related to feature information and structural properties of the input graph, we consider these attributes in separate methodical studies. Let us take a base graph with fixed characteristics (homophily = 0.8; edge density = 0.03; feature SNR = 1.2). Tables 3.2 and 3.3 present the node classification results on several versions of this graph that correspond to assigning it different connectivity matrices (and respective mutations). These matrices define structures of different homophily, while keeping density constant (and vice-versa, for Table 3.2 and Table 3.3, respectively). Analogously, Table 3.4 displays node classification results for versions of the base graph with different feature information, measured by its SNR; results for the respective random connectivity mutations are also shown.

Homophily The inspection of GCN's response to different homophily conditions (Table 3.2) reveals its adequate performance on the most and least homophilic original graphs. While adequate performance in the most heterophilic scenario might seem surprising at first glance, it is not unexpected in our experiment. This behavior relates to the fact that our learning problem only considers two distinguishable types of nodes and has also been recently reported in other works [Ma et al., 2022]. Nodes are able to encode meaningful representations through neighborhood aggregation, despite most of their

Table 3.3: Node classification performance (mean accuracy ± standard deviation) of GCN and MLP (baseline) models on artificial graphs with fixed homophily (0.8) and features (SNR = 1.2) and different edge density (d_e). Results are shown for each original graph G and the respective mutations of G with random connectivity, random feature assignment, and both. Given its feature-only nature, a single MLP result is shown per feature transformation.

	d_e	Structure		MLP (baseline)
		Original	Random	
Original Features	0.003	0.78 ± 0.03	0.63 ± 0.05	0.71 ± 0.03
	0.03	0.81 ± 0.03	0.59 ± 0.03	
	0.15	0.79 ± 0.02	0.57 ± 0.02	
Random Feature Assignment	0.003	0.49 ± 0.06	0.48 ± 0.05	0.49± 0.03
	0.03	0.49 ± 0.03	0.49 ± 0.02	
	0.15	0.52 ± 0.04	0.49 ± 0.04	

neighbors belonging to a different class, due to its consistency. While this outcome may not hold under different conditions (such as some multi-class problems), it draws further attention to the insufficiency of solely resorting to homophily-related assumptions to steer GNN architecture research endeavors, as discussed in our opening section and in some recent works [Ma et al., 2022; Luan et al., 2023].

Similar to the previous highly informative artificial graph, we verify a significant loss of performance when structure and/or feature information of original graphs are destroyed, except for when homophily is close to 0.5 (mediocre performance on the original graph, on-par with the random structure mutation). This means that when nodes are equally likely to be connected to nodes of a different class as to those of the same class, this produces an uninformative structure. A GCN will use this structure to perform local smoothing operations that will decrease feature expressivity and lead to poor node representations. These results reinforce that simply attributing a GCN's poor performance to graph heterophily is insufficient, as even in highly controlled, simple experiments some heterophilous graphs can encode relevant structure information while others do not.

Edge density Varying edge density appears to have a minimal effect in overall performance under the tested conditions (Table 3.3). In theory, we would expect to see better node representations for graphs with fewer connections when structure is irrelevant, as these would lead to a minimal smoothing effect. However, while the average performance for random structure decreases as graphs become more densely connected, these results cannot be deemed statistically significant. This outcome is a good evidence of the intricate relationships between graph properties: manipulating edge density on its own while keeping all other properties constant lead to no significant impact in GNN performance; however this is not necessarily the case when more than one graph property are manipulated at the same time (see Table 2.2). While the focus of this study is to isolate the influences of different properties systematically (and not in combinations), these results evidence that this is another important line of research (further discussed under future work in Section 5).

Varying Feature Signal

Table 3.4: Node classification performance (mean accuracy ± standard deviation) of GCN and MLP (baseline) models on artificial graphs with fixed structure (homophily = 0.8; edge density = 0.03) and different feature SNR. Results are shown for each original graph G and the respective mutations of G with random connectivity.

SNR	Structure		MLP (baseline)
	Original	Random	
1.0	0.57 ± 0.05	0.51 ± 0.03	0.51 ± 0.05
1.2	0.81 ± 0.03	0.59 ± 0.03	0.71 ± 0.03
1.5	0.92 ± 0.02	0.68 ± 0.02	0.91 ± 0.02
2.0	0.99 ± 0.01	0.72 ± 0.01	1.00 ± 0.00

Table 3.4 shows the impact of considering different levels of separability of node features when structure encodes useful information and when it does not. The results indicate a significant loss of performance for the random connectivity mutation in comparison with the original version, even in scenarios when base features are easily separable by a feature-only method. A meaningful graph structure can, however, greatly assist the classification task in cases when features are not easily separable, as we can verify by the significant performance gains of using GCN with the original structure in comparison with the feature-only MLP for SNR $\in [1.0, 1.2]$. This scenario is thus the more evident *sweet spot* for representation learning tasks through graph convolutions, as the next sections will further discuss.

3.2 Feature/Structure Decoupling in Complex Scenarios

Building on the insights obtained from the controlled settings, we now transition to a broader empirical context by applying the same decoupling framework to a range of more complex GNN architectures and standard benchmark datasets. This shift from synthetic to real-world graphs, and from simple GCNs to more sophisticated models, enables us to probe whether the observed interactions between feature and structural signals persist under more complex modeling assumptions and in less idealized data regimes.

In particular, we aim to uncover whether and how distinct GNN designs exhibit different sensitivities to perturbations in input modalities. This broader analysis sets the stage for identifying systematic trends across architectures and datasets, shedding light on the interplay between model assumptions and graph characteristics. As such, besides the baselines, the following GNNs are selected: reversible GCN (RevGCN) [Li et al., 2021], due to its robustness to oversmoothing even for deep GNNs; APPNP [Gasteiger et al., 2018] and FiLM convolution [Brockschmidt, 2020] which seem to perform adequately for input graphs that belong to different parts of the graph properties spectrum proposed in [Palowitch et al., 2022], where vanilla-GCNs do not. This mindful selection of models is intended to give us a diversified view of learning behaviors while keeping a feasible number of runs and analyses. The results are presented in Figure 3.4, where mean and standard deviation are reported for the testset based on 10 independent runs using a fixed 80-10-10% train/validation/test split. Hyperparameter configuration and implementation details are provided in Appendix A.4.

Our experimental results reveal that *different GNN architectures*—each reflecting distinct inductive biases—*navigate the trade-off between preserving raw feature information and leveraging relational structure in markedly different ways* (Figure 3.4). These findings extend the patterns identified in earlier controlled settings, offering new insights into how architectural design influences a model's sensitivity to feature or structure perturbations. The following analysis delves deeper into these dynamics, highlighting how specific design choices shape learning behavior across diverse graph scenarios.

A Feature-Preserving Quality The results depicted in Figure 3.4 show that models that are generally more feature-dependent (i.e., those that respond more to changes of the feature distribution than to structural changes) are particularly robust in handling meaningless structural information and overcoming its potentially harmful effect. However, the same models can also underperform compared to vanilla-GCN for cases in which encoding graph structure through graph convolutions is particularly beneficial (e.g., *Cora* and *CiteSeer* with original structure). This is the case for RevGCN and FiLM. Their feature-preserving quality can be particularly observed in their overlap with MLP performance for random connectivity scenarios (see consistent overlap between RevGCN/FiLM and MLP performance for G_A mutation in Figure 3.4).

A Strong Smoothing Effect GCN and APPNP seem to lack this feature-preserving quality, consistently leading to more smoothed node representations, regardless of whether that smoothing is beneficial for the final task or not. For these models, feature and structure information are both highly important. In cases in which either of these

		Original Structure				Random Connectivity (Mutation A)				
		GCN	RevGCN	APPNP	FiLM	GCN	RevGCN	APPNP	FiLM	MLP
Original Features	Informative*	0.96	0.96	0.95	0.96	0.69	0.92	0.70	0.92	0.92
	Cora	0.78	0.72	0.79	0.67	0.43	0.67	0.48	0.64	0.64
	CiteSeer	0.73	0.72	0.74	0.68	0.45	0.68	0.49	0.67	0.66
	Chameleon	0.39	0.39	0.41	0.40	0.41	0.42	0.46	0.44	0.43
	Texas	0.53	0.54	0.56	0.72	0.49	0.66	0.56	0.73	0.69
Random feature assignment (Mutation B)	Informative*	0.61	0.68	0.54	0.65	0.48	0.49	0.49	0.50	0.50
	Cora	0.57	0.45	0.64	0.39	0.23	0.19	0.25	0.20	0.19
	CiteSeer	0.56	0.40	0.61	0.37	0.20	0.19	0.20	0.20	0.20
	Chameleon	0.23	0.22	0.21	0.21	0.23	0.22	0.22	0.22	0.22
	Texas	0.49	0.51	0.54	0.57	0.51	0.43	0.52	0.54	0.54

* Artificial graph with informative features (SNR=1.5) and structure (homophily=0.95; edge density=0.06)

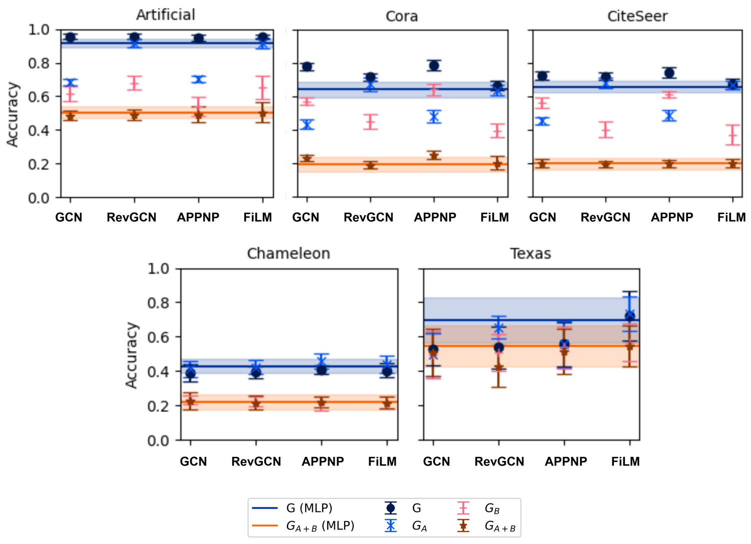


Figure 3.4: Node classification performance (accuracy) of different GNN architectures and MLP (baseline) models on real-world benchmarks. Results are shown in table and graphical form for each original graph G and the respective mutations of G with random connectivity (mutation $A \leftrightarrow$ graph G_A), random feature assignment (mutation $B \leftrightarrow$ graph G_B), and both (graph G_{A+B}). A single MLP (feature-only) result is shown per feature transformation. Shaded areas (MLP) and error bars (GNNs) represent standard deviation.

components holds irrelevant information for the downstream task, we can see significant performance drops compared to the original scenario (see GCN and APPNP performance for G_A and G_B mutations on the *informative artificial graph*, *Cora* and *CiteSeer*).

By decoupling the feature and structural influences on GNN performance, our experiments reveal the nuanced ways in which different levels of graph information impact various architectures on the same classification task. Notably, our findings indicate that advanced architectures—designed to mitigate known limitations of simpler GCNs, such as harmful (over-)smoothing—do not always outperform vanilla methods, as their effectiveness fundamentally depends on the extent to which their architectural assumptions align with the attributes and properties of the input graph.

These insights are enabled by the model-agnostic nature of our proposed method, which quantifies the relative contributions of features and structure to model performance for a given architecture. As a result, our approach facilitates more meaningful comparisons of GNNs, supports the targeted selection of architectures for specific tasks, and enables explicit identification of learning bottlenecks—ultimately guiding the development of more effective graph learning models.

4 Measuring the Interplay between Feature and Structure Components

The results presented in the previous subsections suggest a consistent interdependence between feature and structural components in determining GNN performance. While earlier experiments provided strong empirical indications of this interplay, we now turn to a more formal and quantitative analysis to substantiate these observations. In particular, we ask: *Can the interaction between structure and feature contributions help explain when and why a GNN is more effective than a simpler, feature-agnostic alternative?*

Regression Analysis: Predicting GNN Performance To address this question, we consider the absolute performance gain of a GNN over an MLP baseline—a structure-agnostic model—as a proxy for the overall advantage gained through message-passing and representation learning on graphs. Our goal is to predict this gain using the separate contributions of structure and feature components, as defined by the metrics C_{struct} and C_{feat} introduced in Section 2 (Equations 3.1 and 3.2, respectively).

We hypothesize that these two quantities, and particularly their interaction, can serve as effective predictors of GNN performance gains. To test this, we fit two linear regression models: one using only the independent contributions of structure and features as predictors, and another including an interaction term $C_{\text{struct}} \times C_{\text{feat}}$. The results are

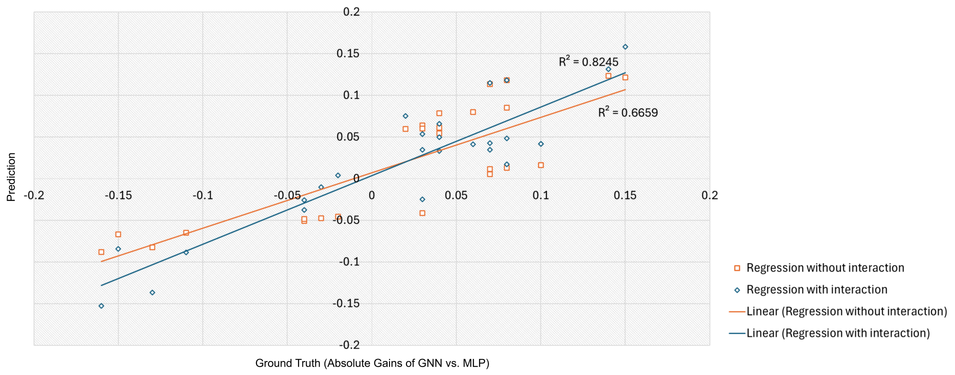


Figure 3.5: Regression plots comparing observed GNN gains over MLP (x-axis) to predicted gains (y-axis), using structure contribution (C_{struct}), feature contribution (C_{feat}), and their interaction as predictors. The model that includes the interaction term $C_{\text{struct}} \times C_{\text{feat}}$ achieves a significantly better fit, suggesting that the effects of structure and feature contributions are not purely additive and that their joint influence plays a key role in determining model performance. Detailed regression statistics are reported in Tables B.5 and B.6 in Appendix B.1.

summarized in Figure 3.5, which plots the actual performance gains (x-axis) against the predicted values from each model (y-axis). As shown, incorporating the interaction term significantly improves the fit ($R^2 = 0.82$), providing strong evidence that the combined effect of structure and features is not merely additive, but interdependent. Further statistical details regarding model fit and significance are available in Appendix B.1 (Tables B.5 and B.6).

Visualizing Feature-Structure Interactions To further unpack the nature of this interaction, Figure 3.6 stratifies the data based on feature contribution. Specifically, we divide dataset+model pairs into two groups: those with C_{feat} above the median ("high feature reliance") and those below the median ("low feature reliance"). Each point represents one such pair, plotted by its structure contribution (C_{struct}, x-axis) and corresponding absolute GNN gain over MLP (y-axis).

For the group with low feature reliance (*orange* markers), we observe a strong positive relationship: the greater the structural contribution, the more the GNN outperforms the MLP. This suggests that in scenarios where features are weak, models can significantly benefit from learning over the graph structure. Conversely, for the high feature reliance group (*blue* markers), the slope is nearly flat, indicating that structural quality has minimal impact—performance remains relatively stable regardless of C_{struct}. This

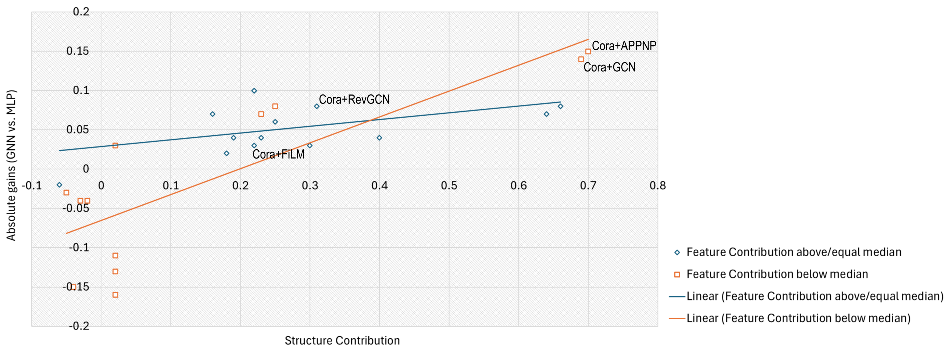


Figure 3.6: Interaction plot illustrating how the relationship between structure contribution and GNN gains over MLP varies with different levels of feature contribution. Data is split based on feature contribution: above-median (*blue*) and below-median (*orange*). A stronger positive relationship is observed in the low feature contribution group, indicating that structure plays a more decisive role when feature reliance is limited.

highlights an important asymmetry: strong features can mask the benefits of graph structure, especially when architectures default to feature-preserving behavior.

Implications for GNN Architecture Design These findings provide novel and quantitative evidence of the complex interplay between feature and structure components in GNNs. In particular, they show that structure only becomes a significant driver of GNN success when feature information is insufficient on its own. This is consistent with observations from previous sections where architectures such as RevGCN and FiLM, which exhibit a stronger tendency to preserve features, underperform in settings where structural information could have been more beneficial (e.g., *Cora* and *CiteSeer*).

More importantly, this trade-off highlights an opportunity for architectural innovation. Current GNNs often implicitly prioritize either feature-preservation or structure-smoothing, but rarely both in a balanced or adaptive manner. Our results suggest that more flexible architectures—perhaps through dynamic attention over feature/structure channels or learnable mixing strategies—could better exploit the full spectrum of graph information. Such models could overcome the current performance ceiling by adapting to the varying information landscape across datasets and tasks.

Altogether, this analysis offers compelling quantitative support for the importance of feature/structure interplay in determining GNN effectiveness. Regression modeling with interaction terms enables a more precise interpretation of how different levels of feature and structural signal combine to affect performance. These insights caution against treating features and structure as isolated drivers of model success and instead advocate for a more integrated view of graph representation learning. In order to move forward, architectural choices should also be guided by empirical alignment with the properties of the input graph data.

5 Discussion

The experiments presented throughout this chapter challenge the common assumption—introduced at the opening section (Section 1)—that message-passing through graph convolution operations inherently produce beneficial smoothing that improves node representations. Our findings reveal that the effectiveness of GNNs critically depends on the alignment between the graph's structural properties, feature informativeness, and the underlying downstream task. When the structure exhibits coherence with class-relevant patterns—such as, but not limited to, homophily—message-passing mechanisms can perform meaningful denoising through local aggregation, enabling GCN-based models to extract enhanced representations that surpass those of feature-only baselines. Conversely, when the graph structure is noisy, weakly informative, or misaligned with label distributions, the smoothing induced by standard GNNs may instead blur important feature distinctions and amplify noise. In such scenarios, the assumption of universally beneficial smoothing fails, leading to situations where GNNs underperform even compared to simple MLPs that ignore structural information altogether. This assumption, while intuitive, is an obvious oversimplification; as we started by showing in Figure 3.2, graph convolution-induced smoothing is not universally beneficial and can sometimes degrade performance. However, starting from this simplified premise enabled us to systematically delve deeper into the complex interactions between multiple factors that are challenging to disentangle—such as (excessive) smoothing, graph topology, feature informativeness, the interplay between feature and structure, and other latent influences on GNN behavior.

Our findings also extend to more advanced GNN architectures. Models such as RevGCN and FiLM are often more robust to noisy or uninformative structure, but we show that this robustness comes from a tendency to preserve feature information and ignore structure entirely when it is not clearly useful. While this avoids performance degradation in low structure informativeness scenarios, it also results in sub-optimal performance when structure *is* informative. These results emphasize the importance of a nuanced, data-driven evaluation of graph structure's contribution in each task, rather than relying on GNNs' structural inductive biases as a default strength.

The method we propose—based on decoupling structure and feature contributions via controlled perturbations—offers a new empirical lens through which to assess how and when GNNs leverage different modalities of graph information. This approach not only helps interpret GNN behavior on a per-graph basis, but also reveals generalizable insights into model design and graph representation learning. The next subsection distills these insights.

5.1 Key Insights

Quantifying the Role of Structure in GNN Learning Our feature/structure decoupling methodology provides a principled, model-agnostic way to quantify the contribution of each component to GNN performance. The ability to isolate and measure these effects on any given task-model pair offers a clear advantage over intuition-driven practices or domain-specific heuristics. We show that the structure contribution metric correlates strongly with GNNs' gains over MLPs when feature information is weak—underscoring the crucial role of structure when features alone do not suffice. Conversely, when feature contribution is high, we observe diminishing returns from structure, even when it is of good quality.

These patterns were consistently observed across architectures and benchmarks. As such, they support a more selective and evidence-based application of graph convolution methods: structure should be leveraged when it offers complementary information, not assumed to be beneficial by default.

Smoothing as Feature Denoising: When GCNs Excel We find that GCNs offer the largest gains over structure-agnostic baselines in graphs where structure enables effective local smoothing of noisy feature signals. This identifies an important sweet spot for vanilla GCNs: moderate feature noise combined with structurally coherent neighborhoods. Our results confirm that GCNs can act as effective feature denoisers under these conditions—validating their theoretical foundations in practice.

Importantly, our approach allows practitioners to identify this regime *a priori*, by measuring the structural and feature contributions before full model deployment. This can guide architecture selection and avoid enforcing structure in graph problems that do not benefit from it.

Revealing Learning Trade-offs in Modern GNN Architectures More sophisticated GNNs such as RevGCN and FiLM are often motivated by the desire to mitigate some learning challenges known to affect GNNs (recall Subsection 3.4 of Chapter 2), such as oversmoothing. Our experiments reveal that these architectures tend to fall back on the feature component when structure is unreliable, resulting in stable, but sometimes unremarkable, performance. This adaptive behavior can be seen as a feature-preserving bias. While beneficial in noisy-structure regimes, it also leads to missed opportunities where structural signals could otherwise enhance performance.

Our regression analysis (Section 4) further supports this finding: the interaction between structure and feature contributions is not purely additive. Instead, structure contributes most when feature signals are weak, and contributes less or not at all when features dominate. This underscores that GNN design is not just about increasing expressivity, but about learning to dynamically weigh multiple information modalities depending on their reliability and relevance.

Guiding Future Model Design Collectively, our findings highlight the need for GNNs that can reason about the relative importance of structure and features during training. Rather than encoding fixed biases toward message passing or raw features, future architectures might benefit from learning mechanisms that dynamically adapt to graph properties. As such, it might be worth it to consider a symbiotic co-attention over structure and feature channels, or structure-aware masking techniques based on *a priori* estimated utility of graph structures.

Our methodology provides the empirical foundation for such future developments. It offers a diagnostic framework that quantifies where existing models succeed or fail, and it helps uncover bottlenecks that may be addressed by more flexible GNN operations. Rather than designing new GNNs in the dark, researchers can now use measurable indicators of feature/structure utility to guide innovation.

5.2 Related Work

Causes of Poor GNN Performance Many authors have investigated poor GNN performance due to apparent oversmoothing or loss of expressivity in deeper GNNs [Li et al., 2018; Cai and Wang, 2020; Balcilar et al., 2021; Oono and Suzuki, 2019; Yan et al., 2022]. This phenomenon is typically explained as deeper networks excessively smoothing the features between distant graph nodes. Others have argued that poor results are not necessarily associated with oversmoothing, but are rather a consequence of the training difficulty of GNNs [Luan et al., 2024; Hu et al., 2019; Cong et al., 2021]. Oversquashing [Topping et al., 2022] and heterophily [Yan et al., 2022] have also been pointed out as causes for performance degradation. Despite their generalized acceptance within the GNN research community, these different causes/concepts are to some extent intertwined and it is difficult to detach one from the others and measure their individual contribution to the performance degradation witnessed downstream.

This fact could be the reason why it is currently possible to find contradictory information in theoretical studies regarding GNN learning behaviors [Keriven, 2022; Cong et al., 2021]. Therefore, there is a demand for relevant empirical studies addressing these questions. In Luan et al. [2023], for example, the authors show how GNN's performance goes beyond homo-/heterophily alone through empirical analyses with dedicated quantitative metrics; however, no relation with feature distribution/quality was explored. Our work aims to fill this gap, bringing forward novel insights on the key conditions for effective GNN learning (both on structural and feature levels) through a dedicated set of experiments under full control of graph attributes.

Advanced GNN Architectures New GNN architectures have emerged in recent years aiming to address specific challenges/limitations of these networks. These advanced architectures propose to address expressivity [Veličković et al., 2018; Xu et al., 2019], computational cost [Hamilton et al., 2017; Li et al., 2021], graph oversmoothing [Chen et al., 2020; Zhao and Akoglu, 2019] and heterophily [Yan et al., 2022; Pei et al., 2019; Bodnar et al., 2022]. While these methods address some of the previous limitations, their empirical gains for the classification task are not always well understood. It can also be difficult to fully compare these approaches, due to the higher complexity, the different base assumptions, and insufficient benchmarking analyses. This makes the identification of learning bottlenecks a non-trivial problem for GNNs. The adoption of more robust model-agnostic, experimental analyses, such as the ones proposed in this work, could help the community better understand these bottlenecks.

Feature/Structure Influence in Graph Learning Decoupling feature and structure impact in GNNs has recently been investigated for transfer learning and graph generation purposes [Thang et al., 2022] and on some types of synthetic graphs [Castellana and Errica, 2023]. However, Thang et al. [2022] assume node homophily and do not explore the cases where this condition is not met, which is one of the aspects in which our work differs. [Castellana and Errica, 2023], on its hand, solely tested on synthetic graphs, drawing conclusions based on assumptions at the moment of graph generation, namely neighborhood feature similarity and class label-aware structural biases; our work presents a method that also allows us to step out of the fully controlled scenario to real-world benchmarks and more complex model architectures, leading to significantly different insights. Other works also explore homo-/heterophilic settings and create advanced architectures to handle challenging scenarios [Zhu et al., 2020]. Nevertheless, it is not clear whether these architectures lead to more useful node representations or if they solely overcome its performance hindering impact, as we can see in recent benchmarks (see Figure 3.1). Our work intends to complement these efforts by providing insights on how structure can not only be harmful but also simply uninformative, leading to local smoothing operations that decrease feature expressivity, in which case one might simply resort to feature-only methods.

5.3 Looking Ahead

This chapter advances our understanding of GNN behavior through a rigorous empirical methodology that disentangles feature and structural contributions. In doing so, it calls into question widespread assumptions and offers concrete paths toward more data-aware model selection and design. As GNNs continue to be applied across increasingly diverse domains, the importance of such empirical grounding will only grow.

Limitations Despite these contributions, our study has several limitations that should be acknowledged. Our results suggest the need to explore not only feature/structure co-dependence but also how models respond to specific combinations of graph properties—an aspect not yet addressed in our controlled experiments. Furthermore, we relied on artificial graphs where the meaningfulness of their information is based on theoretical assumptions. This reliance may limit the empirical conclusions drawn, as we focused primarily on extreme scenarios. Additionally, our method of destroying structural information does not eliminate all graph properties (e.g., edge density remains unchanged). While this should not affect our main conclusions, it may restrict the method's applicability in broader contexts.

Future Work Building on these limitations, future work should aim to deepen our understanding of model responses by extending our approach to more complex scenarios, including simultaneous variation of multiple structural properties alongside feature information. Expanding experiments to encompass a wider range of graph conditions will help validate and refine our assumptions under more realistic settings. Moreover, the feature-structure decoupling method introduced here has the potential to become a more automated empirical tool for assessing how informative a graph's structure is relative to a given network architecture—an area still lacking robust solutions. Beyond analyses, our findings also provide valuable insights to guide the future design of GNN architectures, encouraging models that better balance and leverage feature and structural information. We posit that advancing these directions will be critical for developing GNNs that fully exploit the complex interplay of features and structure in real-world graphs.

6 Conclusion

The interplay between graph learning and deep learning offers both opportunity and complexity, demanding a nuanced understanding of how feature and structural information jointly shape model behavior. Throughout this chapter, we systematically explored the decoupled influence of these components in GCNs and, more broadly, in GNN architectures. By critically examining when GNNs may not outperform feature-only methods, we revealed that graph structure can at times undermine learning, not due to heterophily *per se* but when graph structure and attributes fail to align with the assumptions of the model.

The proposed feature/structure decoupling method provides a practical lens to diagnose and quantify the contributions of each component, thereby offering valuable insights into identifying learning bottlenecks. Our empirical evidence underscores that GCNs can produce poor node representations even in the absence of oversmoothing if the input graph's structure does not suit their local smoothing bias. Moreover, advanced GNN architectures are often found to revert to feature-preserving encodings akin to feature-only baselines when they cannot effectively leverage structural information, despite their more sophisticated aggregation schemes.

Collectively, these findings challenge the notion of GNNs as a one-size-fits-all solution, instead highlighting the need for task-specific design choices and graph analysis prior to deployment. As the field progresses, bridging the gap between theoretical insights and empirical performance will be crucial for the development of robust and effective graph learning models. This chapter leaves concrete, practical insights for optimizing both learning objectives and network architectures, enabling practitioners to align model design with the intricacies of the graph structure and the task at hand, ultimately driving performance gains where they are most impactful.

4 The Illusion of Depth Scaling in Graph Neural Networks

GNNs have demonstrated substantial potential in learning representations from graph-structured data for several applications, provided they appropriately leverage the interplay between node features and topological information. Yet, despite their promising capabilities, fundamental challenges remain in defining the optimal architectural and design choices for GNNs to ensure robust, efficient, and scalable learning. One particularly contentious design aspect is the *role of depth scaling*: how the addition of more layers influences the representational capacity and practical performance of GNNs. This chapter adopts a holistic view to explore depth scaling in GNNs, starting with a overview of state-of-the-art methods and their meticulous analysis and proceeding to a dedicated empirical study on how these networks learn, with the aim of clarifying its implications for architecture design and providing actionable insights for the development of more effective graph learning systems.

Section 1 formalizes the problem and situates it within the broader context of representation learning on graphs. Then, the chapter proceeds to provide a critical review of existing architectural strategies and depth scaling recipes for GNNs (Section 2), complemented by an in-depth case study evaluating the discriminative power of a specific deep GNN architecture, GCN+InitRes (Section 2.2). This analysis is followed by the introduction of a *novel depth evaluation protocol for GNNs*, addressing the limitations of existing approaches and exposing two types of pathological learning behaviors in deep GNNs (Section 3).

Through this comprehensive methodology, the chapter establishes several important findings: that depth scaling can lead to redundant representations and noisy or uninformative layers in GNNs; that the search space for GNN architectures can often be narrowed to shallow networks, promoting computational efficiency without sacrificing expressivity; and that certain architectural enhancements, such as residual connections, can improve the discriminative power of the network in a task-dependent manner. These insights are synthesized in the comprehensive discussion of Section 4, which contextualizes the *broader implications of these findings for GNN architecture design and practical graph learning applications*.

Finally, the chapter concludes with a summary that encapsulates the contributions and highlights future directions for addressing the persistent challenges of depth scaling and architectural optimization in GNNs (Section 5). In this way, the chapter contributes to the overarching goal of *enhancing representation learning with GNNs by advancing our understanding of their behavior under increasing depth and refining design choices to build more effective models*.

Remark

The main contents of this chapter were published under the following venue:

Gomes, D., Efthymiadis, K., Nowe, A., & Vrancx, P. (2024). Depth Scaling in Graph Neural Networks: Understanding the Flat Curve Behavior. *Transactions on Machine Learning Research (TMLR).*

1 Problem Statement

Deep GNNs show very different trends from other deep learning methods. Unlike conventional neural networks used in other domains (e.g. CNNs in computer vision), for which increasing the number of stacked layers is generally associated with greater expressive power and improved performance, the basic versions of GNNs (such as GCN [Kipf and Welling, 2017] or SGC [Wu et al., 2019]) do not benefit from depth; on the contrary, increasing depth leads to a significant drop of performance (see GCN in Figure 4.1). This trend is generally explained by the oversmoothing problem—a phenomenon in which all nodes in a graph become indistinguishable as a consequence of multiple aggregations of node representations through message-passing operations, as we have seen in the previous chapters.

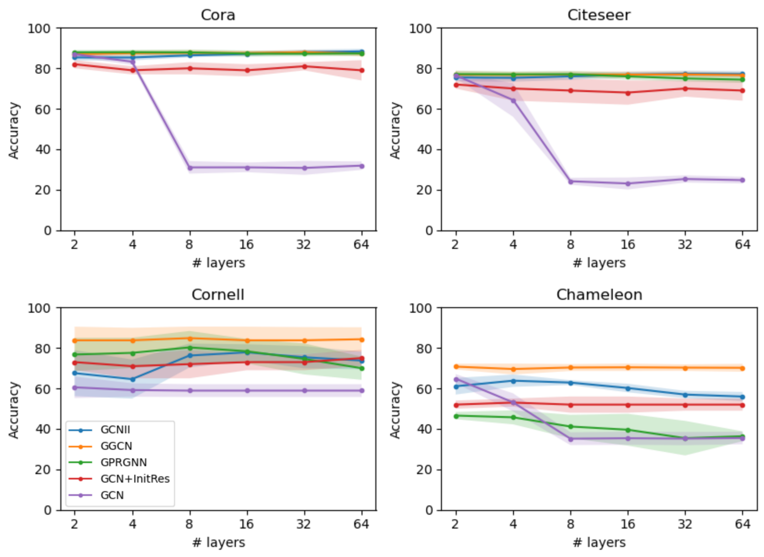


Figure 4.1: Examples of the *flat curve* behavior (performance stagnation over depth in GNNs). Results for GCNII, GGCN, GPRGNN and GCN were extracted from Yan et al. [2022]. GCN+InitRes refers to results of this work (for $\alpha = 0.5$).

At the same time, the motivation to train deep GNNs still stands, as a k-layer GNN could potentially capture useful long-range dependencies by enabling the aggregation of relevant information from nodes up to k hops away. This follows from the standard message-passing paradigm, in which each layer updates a node's representation by aggregating features from its immediate neighbors. As more layers are stacked, this local aggregation process compounds, allowing information to propagate across larger portions of the graph. Thus, a k-layer GNN theoretically enables each node to integrate signals from its k-hop neighborhood, thereby expanding its receptive field and allowing it to model more complex relational patterns and long-distance interactions. This observation has led to the proposition of more complex network architectures, which do enable us to train deep networks to solve challenging tasks with adequate performance (e.g., GCN with initial residual [Gasteiger et al., 2018; Chen et al., 2020; Jaiswal et al., 2022], GCNII [Chen et al., 2020], GGCN [Yan et al., 2022], GPRGNN [Chien et al., 2020], G^2 [Rusch et al., 2023b]). However, these deep architectures often lead to equivalent or even deteriorated performance when compared to their shallow counterparts, as Figure 4.1 clearly shows. This opens unresolved questions regarding the pursuit of depth in GNNs.

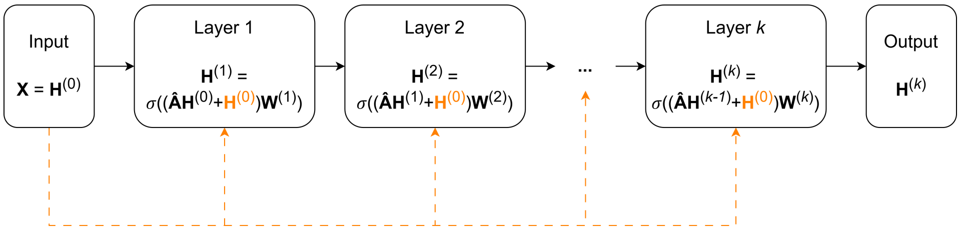

Input
X = H(0)
Layer 1
H(1) = σ((ÂH(0)+H(0))W(1))
Layer 2
H(2) = σ((ÂH(1)+H(0))W(2))
...
Layer k
H(k) = σ((ÂH(k-1)+H(0))W(k))
Output
H(k)

Figure 4.2: (Simplified) Architecture of a deep GCN with initial residual connections from the input features $\mathbf{X}^{(0)} = \mathbf{H}^{(0)}$ to each hidden layer. Each layer aggregates information from the previous layer via the normalized adjacency matrix $\hat{\mathbf{A}}$, while also incorporating the original input (depicted in *orange*).

The Controversial Role of Depth in GNN Learning Previous works have elaborated on the theoretical role of depth, but their conclusions often lack further empirical validation, and it is even possible to find contradictory arguments regarding the potential benefits of depth following theoretical reasoning [Keriven, 2022; Cong et al., 2021]. Poor GNN performance due to apparent oversmoothing or loss of expressivity in deeper GNNs has also been investigated in some works [Yan et al., 2022; Li et al., 2018; Balcilar et al., 2021; Keriven, 2022; Oono and Suzuki, 2019]; but, as we have seen in Chapter 3 and supported by further very recent works [Arnaiz-Rodriguez and Errica, 2025], these considerations are often insufficiently justified and the studied phenomena are entangled. This type of confusion has led other authors to argue that it is not oversmoothing, but rather the training difficulty of GNNs that leads to poor results [Luan et al., 2024; Cong et al., 2021]. This scenario evidences that the way that GNNs learn is still poorly understood and constitutes the main motivation for our study, as it shows that clear and dedicated empirical analyses on GNN learning behavior at the face of increasing depth are in demand.

In this chapter, we aim to *explain the plateau of performance over depth* observed in Figure 4.1 by conducting a comprehensive empirical study across eleven widely used benchmark datasets for semi-supervised node classification. We employ GCN with initial residual connections (GCN+InitRes, Figure 4.2) as a representative case study, since this architecture exhibits performance trends that are analogous to those of the remaining, more complex models while pertaining a layer expression with interesting properties to anchor our analyses. Our investigation first quantifies the extent to which the flat performance curve emerges across different depths, before turning to an in-depth examination of the intermediate hidden node representations in several state-of-the-art deep GNN architectures. This examination seeks to uncover the underlying factors driving the observed plateau and to provide empirical justification for the phenomenon. All experiments are carefully crafted under an empirically driven mindset, while being theoretically grounded and meticulously justified and framed within recent research findings, ensuring that the insights drawn are both rigorous and relevant to the ongoing discourse on GNN depth scaling.

2 Deep Graph Neural Networks

Inspired by the success of deep learning in other domains, searching for GNN layers that can scale with respect to depth has been a driver for researchers in the field, despite well-known bottlenecks. Among these bottlenecks, there is oversmoothing—particularly important in this context since a network that produces oversmoothed node representations inevitably decreases its discriminative power to the point of random classification. As such, pursuing depth in GNNs inherently assumes that the network architecture accommodates at least a method enabling it to avoid this phenomenon.

Throughout this section, we explore what these methods are, how they are combined and what happens when we increasingly stack layers to create deep architectures.

2.1 Current Recipes for Depth Scaling

Let us start by examining current methods and how these have been combined in layers that enable deep architectures. We further discuss what these approaches have in common and what may distinguish them.

Residual Connections

Residual connections are one of the most commonly employed methods to mitigate oversmoothing, thus enabling the training of deep GNNs. The terms *residual* and *skip* connections are often used interchangeably to refer to a family of methods that enable the flow of information directly from representations of previous layers to later, deeper ones in deep neural networks. If we stay closer to the initial proposition of residual connections [He et al., 2016], we can define them by Equation 4.1, as also proposed in the original GCN work [Kipf and Welling, 2017] with the purpose of containing oversmoothing and the inherent performance drop associated with increasing depth. In this equation, $\hat{\mathbf{A}}$ represents the normalized adjacency matrix (with added self-loops), $\mathbf{H}^{(l)}$ represents the node embeddings at layer l, $\mathbf{W}^{(l)}$ corresponds to a matrix of learned weights, and σ is an activation function.

$$\mathbf{H}^{(l)} = \sigma(\hat{\mathbf{A}}\mathbf{H}^{(l-1)}\mathbf{W}^{(l)}) + \mathbf{H}^{(l-1)} \tag{4.1}$$

However, in its simplest form, this method only seems to be moderately effective, as results show that oversmoothing tends to be delayed but not fully prevented and we should expect a performance drop as we stack more layers. As such, subsequent works [Chen et al., 2020; Gasteiger et al., 2018] proposed a direct connection to the initial node representation in the form of Equation 4.2, where $\alpha \in [0, 1]$ is a parameter that controls the strength of the residual connection.

$$\mathbf{H}^{(l)} = \sigma(((1-\alpha)\hat{\mathbf{A}}\mathbf{H}^{(l-1)} + \alpha\mathbf{H}^{(0)})\mathbf{W}^{(l)}) \tag{4.2}$$

Other connections (dense [Guo et al., 2019], jumping knowledge [Xu et al., 2018]) have also been proposed, but these have hardly managed to prevent the performance drop trend [Jaiswal et al., 2022]. The initial residual connection defined by Equation 4.2 (GCN+InitRes), on the other hand, has been consistently assisting the training of deep GNNs without significant performance loss, which has motivated their adoption (either as described or with subtle adaptations) in recent works [Jaiswal et al., 2022; Kulatilleke et al., 2022; Feng et al., 2023; Liu et al., 2021; Zhang et al., 2022].

Other Methods

Several other methods have been proposed with the purpose of mitigating/reducing oversmoothing, such as *normalization and regularization* techniques [Zhao and Akoglu, 2019; Rong et al., 2020]. Others use more complex layers and have been broadly categorized as *architectural tweaks* [Jaiswal et al., 2022] or *changing GNN dynamics* [Rusch et al., 2023a]. Methods such as GraphCON [Rusch et al., 2022], GRAFF [Giovanni et al., 2023], G^2 [Rusch et al., 2023b], GGCN [Yan et al., 2022], GPRGNN [Chien et al., 2020] fall under this category. These methods frequently combine several other strategies into the same expression, including residual connections which have been deemed necessary to avoid their oversmoothing by Rusch et al. [2023b].

Comparison of State-of-the-art Deep Architectures

Table 4.1 shows examples of layers that have been used to create deep architectures while attaining adequate performance. Despite the different levels of complexity of these layers, one can observe that they share some *key ingredients*:

- **Preserve Earlier Representations:** A way to preserve the information of earlier node representations in later, deeper ones, either by adding a residual connection to $\mathbf{H}^{l-1}$ (GGCN, G^2), an initial residual connection to $\mathbf{H}^0$ (GCN+InitRes, GCNII), or by weighting all intermediate representations in the final embedding (GPRGNN).
- **Weight Graph Convolution Term(s):** Coefficients (learnable or not) that weigh the graph convolution term(s), and, thus, the effect of the increasingly smoothed representations in the latter embeddings.

Table 4.1: Description of layers that enable deep GNN architectures. All layers include (1) a method that enables the preservation of previous node representations (e.g. residual connection); and (2) weighting coefficient(s) for the graph convolution operation term(s).

Method	**Layer**	
GCN+InitRes		$\mathbf{H}^l = \sigma\Big(\Big((1-\alpha)\hat{\hat{\mathbf{A}}}\mathbf{H}^{l-1} + \alpha\mathbf{H}^0\Big)\mathbf{W}^l\Big)$
	α	Scalar (hyperparameter)
GCNII		$\mathbf{H}^l = \sigma\Big(((1-\alpha^l)\dot{\mathbf{A}}\mathbf{H}^{l-1} + \alpha^l\mathbf{H}^0)((1-\beta^l)\mathbf{I} + \beta^l\mathbf{W}^l) \Big)$
[Chen et al., 2020]	α^l	Scalar (hyperparameter)
	β^l	Scalar (hyperparameter)
GPRGNN		$\mathbf{H}^L = \sum_{l=0}^{L} \gamma_l \dot{\mathbf{A}}\mathbf{H}^{l-1}$
[Chien et al., 2020]	γ_l	Scalar (learned)
GGCN		$\mathbf{H}^l = \mathbf{H}^{l-1} + \eta\Big(\sigma\Big(\alpha^l(\beta_0^l\hat{\mathbf{H}}^{l-1} + \beta_1^l(\mathbf{S}_{pos}^l \odot \dot{\mathbf{A}} \odot \mathbf{T}^l)\hat{\mathbf{H}}^{l-1} + \beta_2^l(\mathbf{S}_{neg}^l \odot \dot{\mathbf{A}} \odot \mathbf{T}^l)\hat{\mathbf{H}}^{l-1}\Big)\Big)$
[Yan et al., 2022]	η	Decay function (based on hyperparameters)
	α^l	Scalar (learned)
	β_0^l	Scalar (learned)
	$\hat{\mathbf{H}}^{l-1}$	$\mathbf{H}^{l-1}\mathbf{W}^{l-1} + \mathbf{b}^{l-1}$, $\mathbf{b}$ (bias) $\in \mathbb{R}^{n\times m}$
	β_1^l	Scalar (learned)
	$\mathbf{S}_{pos}^l$	Positive messages matrix, $\mathbb{R}^{n\times n}$
	$\mathbf{T}^l$	Degree correction matrix, $\mathbb{R}^{n\times n}$
	β_2^l	Scalar (learned)
	$\mathbf{S}_{neg}^l$	Negative messages matrix, $\mathbb{R}^{n\times n}$
G^2		$\mathbf{H}^l = (1-\tau^l) \odot \mathbf{H}^{l-1} + \tau^l \odot \sigma\left(\dot{\mathbf{A}}\mathbf{H}^{l-1}\mathbf{W}^l\right)$
[Rusch et al., 2023b]	τ^l	Rates matrix (learned), $\mathbb{R}^{n\times m}$
	l	Layer
	n	Number of nodes
	m	Number of channels
	$\dot{\mathbf{A}}$	Normalized adjacency matrix, $\mathbb{R}^{n\times n}$
	$\mathbf{H}^l$	Hidden embeddings matrix of the l-th layer, $\mathbb{R}^{n\times m}$
	$\mathbf{W}^l$	Weights matrix of the l-th layer, $\mathbb{R}^{m\times m}$
	σ	Activation function (e.g., ReLU, softmax)

These components can be clearly identified in specific layer designs. For instance, in GCNII, the update rule includes both an initial residual connection (via $\alpha^l\mathbf{H}^0$) and weighted control of the graph convolution and transformation terms (via α^l and β^l). Similarly, the update rule of G^2 blends the previous representation with the graph-convolved output (residual connection), and uses the coefficient τ^l to control the degree of smoothing and information incorporation at each step.

By grouping the surveyed strategies into these categories, understanding their effect becomes more straightforward: while preserving earlier representations helps preventing oversmoothing by having a sharpening effect in the later, more smoothed node representations, weighting graph convolution term(s) has the potential to hold back the smoothing effect of the graph convolutions and decrease the speed of convergence to a constant value that defines oversmoothing. Continuous GNN approaches, such as GraphCON [Rusch et al., 2022] and GRAFF [Giovanni et al., 2023], were left out of the scope of this analysis for conciseness, but similar extrapolations could be performed for these architectures, which notably also include a term for preserving earlier node representations and smoothing control coefficients (see Δt for GraphCON and τ for GRAFF in the original publications).

Besides the choice of these elements, we can also observe several unique terms/coefficients in the surveyed layers, which can make them more or less expressive for different tasks, and lead to the offsets in accuracy between different models that we can observe in Figure 4.1. However, it is important to acknowledge that the performance stagnation over depth phenomenon can still be verified, for the simplest of the models and the most complex ones alike, despite the architectural tweaks which add to their complexity.

2.2 On the discriminative power of deep GNN: a case study for GCN+InitRes

The works presented in Table 4.1 show that we *can* design deep GNN architectures; but are there actual empirical gains in going deeper? And can we measure them? In this subsection, we conduct a series of analyses using the simplest of deep GNN models (GCN+InitRes) to provide further and more thorough insights on the potential of depth as a means to increase the discriminative power of GNNs.

GCN+InitRes layers can be defined by a graph smoothing term, equivalent to that of vanilla-GCNs, and an initial feature encoding term (the residual connection) with a sharpening effect. The influence of these terms in the layer-wise node representations is weighted by α and its symmetric (Equation 4.2), i.e., for $\alpha = 0$ we have the vanilla-GCN and for $\alpha = 0.5$ the smoothed representations have the same weight as the initial ones. We will expand on the problem of performance stagnation over depth for GNNs by exploiting GCN+InitRes properties, as the simplest of the deep GNN examples. We aim to find the practical benefits of scaling these models with respect to depth, while enhancing these empirical insights through comparisons with the base models that compose GCN+InitRes (i.e., GCN and MLP).

Methodology We investigate the effect of using initial residual connections to train deep GCNs by conducting a thorough ablation study of the depth (L) and residual connection strength (α). To that end, we conceive a framework, taking L and α as input variables, where each layer assumes the form of Equation 4.2. We also consider a shallow vanilla-GCN and a 2-layer MLP as baselines for performance comparison. We use eleven of the most common dataset benchmarks, including homophilic and heterophilic datasets (more benchmarking and training details in Appendix A.5). The results of the full ablation study are shown in Figure 4.3; Table 4.2 shows a compact version with the results for the best α for each network depth.

Influence of Smoothing/Sharpening Components in GCN+InitRes Table 4.2 enables a clear comparison of best performance for each model type, also considering the two baselines (simple GCN vs. feature-only MLP). We considered the same number of hidden dimensions for all networks, meaning that the number of parameters of each network varies only with depth. For GCN+InitRes, we ran a grid search of α and depth and report the results for the best performing (α, depth) pair. For vanilla-GCN, this means that Table 4.2 shows results for 1- or 2-layers networks for all benchmarks, with the exception of Texas (4-layer GCN was reported because the absolute average of accuracy was superior; however, no statistically significant difference was found compared to the 2-layers version). For the MLPs, we considered networks of 2-layers, as per most frequently seen in the literature in benchmarking analysis with these datasets.

Comparing GCN+InitRes with the baseline models in Table 4.2, we can identify three distinct patterns within our benchmark datasets. *Case 1* groups benchmarks for which GCN+InitRes shows no evidence of superiority when compared to a vanilla-GCN. For these cases, a vanilla-GCN is able to deliver an equivalent or superior outcome compared to that of a GCN+InitRes, regardless of its depth. As such, adding a sharpening component in the form of an initial residual connection seems not to bring an evident empirical benefit in these cases—we call this a *convolution-sufficient regime. Case 2* evidences benchmarks for which GCN+InitRes is consistently superior to its basic components alone. For these benchmarks, the combination of a smoothing and sharpening component in a single model is able to marginally improve the results, thus evidencing a benefit of practical utility—*complementary regime.* This benefit, however, seems to be independent of depth, as both shallow and deep GCN+InitRes models were able to deliver top performance of classification. Concerning *Case 3* benchmarks, we verify that GCN+InitRes achieves top performance for all $L \geq 2$, showing equivalence to the performance of a feature-only MLP in this case, in what we call a *feature-dominant regime.*

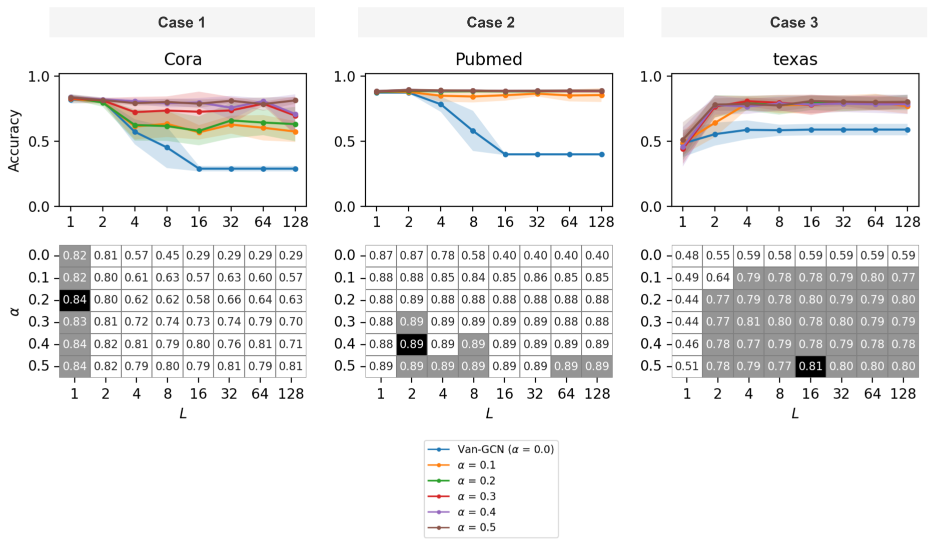
Table 4.2: Node classification accuracy of GCN+InitRes of increasing depth (L) and Vanilla-GCN and MLP baselines for eleven benchmarks. **Bold** values correspond to top performance for each benchmark; Underlined values show no s.s.d. from top performance on each benchmark (row-wise).

	Baselines		**GCN+InitRes**							
	Van-GCN	**MLP**	$L = 1$	$L = 2$	$L = 4$	$L = 8$	$L = 16$	$L = 32$	$L = 64$	$L = 128$
Case 1—Convolution-Sufficient Regime										
Citeseer	.73±.02	.68±.06	**.74±.02**	.72±.02	.70±.06	.69±.06	.68±.06	.70±.04	.69±.05	.71±.03
Squirrel	**.44±.01**	.33±.02	.39±.02	.41±.02	.41±.02	.42±.02	.42±.02	.42±.01	.42±.01	.42±.02
Chameleon	**.60±.03**	.50±.03	.49±.03	.55±.03	.54±.03	.54±.03	.53±.03	.53±.03	.54±.02	.54±.03
Cora	.82±.03	.68±.05	**.84±.02**	.82±.01	.81±.02	.80±.03	.80±.03	.81±.02	.81±.02	.81±.03
Physics	**.97±.00**	.95±.03	**.97±.00**	**.97±.00**	**.97±.00**	**.97±.00**	**.97±.00**	**.97±.00**	**.97±.00**	**.97±.00**
Case 2—Complementary Regime										
Pubmed	.87±.01	.86±.02	**.89±.01**	**.89±.01**	**.89±.01**	**.89±.01**	**.89±.01**	**.89±.01**	**.89±.01**	**.89±.01**
CS	.93±.00	.91±.03	**.95±.00**	**.95±.00**	**.95±.00**	**.95±.00**	**.95±.00**	**.95±.01**	**.95±.01**	**.95±.00**
Case 3—Feature-Dominant Regime										
Cornell	.43±.06	**.75±.02**	.42±.05	.73±.04	.73±.06	.74±.05	.73±.04	**.75±.04**	.75±.04	.74±.05
Texas	.59±.07	**.82±.04**	.51±.13	.78±.05	.81±.04	.80±.06	.81±.05	.80±.03	.80±.06	.80±.05
Wisconsin	.51±.08	**.87±.02**	.59±.05	.85±.05	.86±.03	.84±.03	.84±.04	.84±.03	.84±.05	.84±.03
Actor	.27±.01	.35±.04	.34±.01	.36±.01	**.36±.01**	.35±.01	.35±.01	.35±.01	.35±.01	.35±.01

[*Case 1*] Vanilla-GCN performance is superior or equivalent to GCN+InitRes.
[*Case 2*] GCN+InitRes performs better than both baselines.
[*Case 3*] MLP performance is superior or equivalent to GCN+InitRes.

Ablation Study on Residual Strength and Depth Figure 4.3 shows the performance over depth for different values of α in two views: 1) a plot chart of mean accuracy ± standard deviation over number of layers; 2) a matrix of accuracies per $\langle L, \alpha \rangle$ network, our search space. We conduct paired t-tests to find evidence of statistically significant differences (s.s.d.) of performance between each $\langle L, \alpha \rangle$ and the architecture leading to the best absolute mean accuracy for each dataset. Shaded areas correspond to architectures for which we could not reject the null hypothesis (i.e., no s.s.d. from best performance; more details on the statistical tests can be found in Appendix A.5).

The chart view evidences the *flat curve* behavior that we aim to explain. We can further observe a relation between the magnitude of α and the performance drop to



Case 1 — Cora

α \ L	1	2	4	8	16	32	64	128
0.0	0.82	0.81	0.57	0.45	0.29	0.29	0.29	0.29
0.1	0.82	0.80	0.61	0.63	0.57	0.63	0.60	0.57
0.2	0.84	0.80	0.62	0.62	0.58	0.66	0.64	0.63
0.3	0.83	0.81	0.72	0.74	0.73	0.74	0.79	0.70
0.4	0.84	0.82	0.81	0.79	0.80	0.76	0.81	0.71
0.5	0.84	0.82	0.79	0.80	0.79	0.81	0.79	0.81

Case 2 — Pubmed

α \ L	1	2	4	8	16	32	64	128
0.0	0.87	0.87	0.78	0.58	0.40	0.40	0.40	0.40
0.1	0.88	0.88	0.85	0.84	0.85	0.86	0.85	0.85
0.2	0.88	0.89	0.88	0.88	0.88	0.88	0.88	0.88
0.3	0.88	0.89	0.89	0.89	0.89	0.89	0.88	0.88
0.4	0.88	0.89	0.89	0.89	0.89	0.89	0.89	0.89
0.5	0.89	0.89	0.89	0.89	0.89	0.89	0.89	0.89

Case 3 — texas

α \ L	1	2	4	8	16	32	64	128
0.0	0.48	0.55	0.59	0.58	0.59	0.59	0.59	0.59
0.1	0.49	0.64	0.79	0.78	0.78	0.79	0.80	0.77
0.2	0.44	0.77	0.79	0.78	0.80	0.79	0.79	0.80
0.3	0.44	0.77	0.81	0.80	0.78	0.80	0.79	0.79
0.4	0.46	0.78	0.77	0.79	0.78	0.79	0.78	0.78
0.5	0.51	0.78	0.79	0.77	0.81	0.80	0.80	0.80

Figure 4.3: Results of the ablation study of depth (L) vs. residual connection strength (α) in terms of node classification accuracy for a benchmark example of each case. [Plots] Mean test accuracy over depth per studied α (faded areas correspond to standard deviation). [Performance matrices] Mean accuracy for each $\langle L, \alpha \rangle$ architecture: black shading corresponds to top performance; grey shaded areas show no s.s.d. from top performance.

a semi-constant accuracy range over depth, particularly evident for some benchmarks (e.g. *Cora*). This proves that the behavior under analysis is directly tied to our smoothing/sharpening weighting coefficient (α), which in this case corresponds to controlling the initial residual connection strength. At the same time, we observe equivalent accuracy ranges for all α larger than a certain, dataset-specific threshold, especially for deeper networks.

We can also get a better grasp of the different patterns of the three cases with respect to α by analyzing the matrix view. *Case 1* strongly benefits from the smoothing component, which can even be optimal after a single graph convolution, regardless of the value of α, which can even be $\alpha = 0$. *Case 2* benchmarks benefit from strong residual components ($\alpha \geq 0.3$); and *Case 3* benchmarks achieve top performance for any $\alpha \geq 0.1$. Inspecting the depth dimension, we can once again observe that shallow versions are mostly preferred in *Case 1*, while top performing architectures for *Cases 2* and *3* are mostly independent of depth.

Deep GCN+InitRes are (at best) as discriminative as their shallow counterparts Overall, our results support that, while GCN+InitRes can be more discriminative than the base models that compose it (*Case 2*), *using this method as a means to pursue depth shows no empirical benefit in any scenario.* In fact, we find that there is always at least one shallow GCN+InitRes network (up to 2 layers) able to reach top performance for the studied benchmarks.

3 Depth Evaluation

Benchmark results such as the ones disclosed in the previous case study make us question how we have been evaluating depth for GNNs. Works that conduct analyses on deep versions of new GNNs have been mostly relying on measuring whether they can successfully avoid oversmoothing, but have hardly evaluated whether progressively stacking more layers actually increases the discriminative power of the network. Depth scaling claims have been frequently supported by insufficient benchmarking tables including a single performance metric, frequently accuracy (mean and standard deviation), and consider a method to be superior if it exhibits the highest mean absolute accuracy, even if they have high, overlapping standard deviations and marginal performance gains [Chen et al., 2020; Bodnar et al., 2022; Rusch et al., 2023b].

This section introduces some background on previous depth evaluation approaches and where they fail. We further propose a new protocol that addresses some of their limitations, based on which we disclose a set of learning inefficiencies in state-of-the-art deep GNNs.

3.1 Insufficiency of Oversmoothing Measures

Previous works have evaluated deep GNNs' capacity to overcome oversmoothing by studying the Dirichlet energy [Jaiswal et al., 2022; Rusch et al., 2023a]. Dirichlet energy can be given by Equation 4.3 and can essentially be described as a metric of local node similarity which satisfies useful conditions, namely if $\mathbf{X}_u$ is constant for all nodes $u \in \mathcal{V}$, $\mathbb{E}(\mathbf{X}) = 0$. As such, the layer-wise exponential convergence of $\mathbb{E}(\mathbf{X})$ to 0 defines oversmoothing with respect to this measure.

$$\mathbb{E}(\mathbf{X}) = \sqrt{\frac{1}{|\mathcal{V}|} \sum_{u \in \mathcal{V}} \sum_{v \in N_u} \|\mathbf{X}_u - \mathbf{X}_v\|_2^2} \tag{4.3}$$

Nonetheless, to evaluate the need for depth, it is not sufficient to just evaluate whether we are avoiding oversmoothing. In fact, according to Rusch et al. [2023a], the simple addition of a bias in each layer is enough to keep the Dirichlet energy from exponentially dropping to zero as we increase depth; however, this does not directly imply that we are gaining expressivity and deriving more meaningful node representations by stacking more layers.

3.2 Finding Pathological Behavior in Deep Networks

Central Kernel Alignment (CKA) was introduced by Kornblith et al. [2019] as a robust measure of the relationship between feature representations within the same network and across different networks. In particular, the authors show that CKA (Equation 4.4) is invariant to orthogonal transform and isotropic scaling and can be used to understand network architectures, revealing when depth can become pathological in neural networks representations (e.g., when a big part of the network produces representations that are very similar and, thus, redundant). Following its introduction, CKA has been extensively used to study how deep neural networks learn representations [Nguyen et al., 2021; Raghu et al., 2021], although never to understand the specific case of deep GNNs.

$$\textit{Linear CKA}(\mathbf{H}^l, \mathbf{H}^k) = \frac{||{\mathbf{H}^k}^T \mathbf{H}^l||_F^2}{||{\mathbf{H}^l}^T \mathbf{H}^l||_F \cdot ||{\mathbf{H}^k}^T \mathbf{H}^k||_F} \tag{4.4}$$

Recently, researchers have highlighted certain limitations of CKA, arguing that its interpretation can be counterintuitive [Davari et al., 2022]. Thus, it can be prudent to complement it with other methods that were previously employed to measure representation similarity, such as linear regression. Such methods have been used to validate CKA in its original work, and we argue that they can lead to complementary information towards understanding how deep networks are learning.

3.3 Depth Evaluation Protocol for GNN

In order to study the representations learned by GNNs over depth, we propose a protocol that inspects the layer-wise hidden node representations with two different measures of similarity—Logistic Regression and linear CKA:

1. Train a Logistic Regression classifier on the intermediate hidden representations $\mathbf{H}^l \in \{1, ..., L\}$ of the same network, as a metric of the linear separability of the clusters at each consecutive layer;

2. Compute CKA to measure the pairwise similarity of hidden node representations of layers $l, k \in \{1, ..., L\}$ within the same GNN.

Ideally, hidden node representations of consecutive layers should lead to increasingly more separable clusters (evidenced by higher accuracies if classified by Logistic Regression models) and progressively become more dissimilar from the earlier node

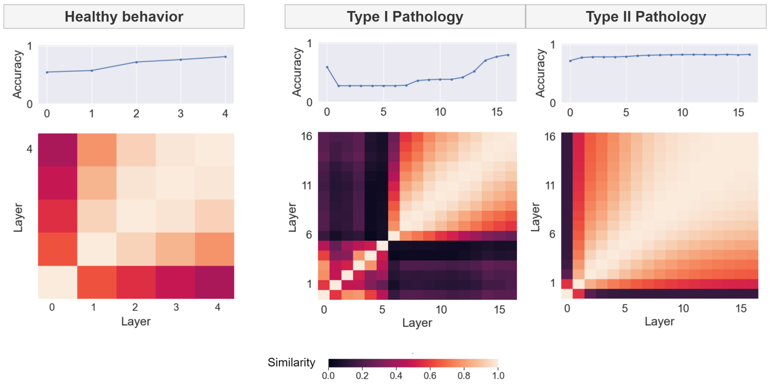


Figure 4.4: Healthy vs. pathological learning behaviors (examples). Layer-wise Logistic Regression is measured in terms of accuracy of classification using the intermediate hidden representations $\mathbf{H}^l \in \{1, ..., L\}$; pairwise CKA of layers $l, k \in \{1, ..., L\}$ is represented by the similarity matrices. Type I Pathology manifests as a sequence of noisy layers (particularly evident for $l < 6$); Type II Pathology manifests as a sequence of nearly identical layers (particularly evident for $l > 10$).

representations of more shallow layers, as stacking more layers is expected to increase the discriminative power of the network. For concision, we will call this a *healthy* behavior (Figure 4.4); a deviation from this trend will be referred to as a *pathology*—a term already coined by previous works [Kornblith et al., 2019] for other types of deep learning models.

Kornblith et al. [2019] used Logistic Regression classification as a reference to validate CKA's efficacy in detecting depth-related pathologies. In this case, we propose an analogous protocol, in which analysing the linear separability of the clusters over depth is not irrelevant/redundant, but complementary. That is because, not only are we interested in identifying cases in which hidden embeddings might show high similarity (and thus be deemed redundant), but also whether we are progressively increasing the discriminative power of the network by stacking more layers. This latter analysis cannot be fully given by CKA alone (e.g. if $\mathbf{H}^l$ and $\mathbf{H}^{l+1}$ evidence low similarity, we cannot know if this dissimilarity is beneficial or not without the assistance of a complementary method able to show whether the classes are becoming progressively more separable).

3.4 Pathological Depth in GNN

We apply our depth evaluation protocol to three deep GNN models: GCN+InitRes, GCNII, G^2. Our experiments intend to verify the hypothesis of pathologic behavior in deep GNNs as a justification for the *flat curve* phenomenon. We show the results for one benchmark of each case surveyed in Table 4.2, to remain succinct while showing evidence of different patterns of behaviors. Further extensive results can be found in Appendix B.2 for all the remaining benchmarks, along with further analyses/visualizations of severe cases, large-scale datasets and attentional GNNs (GAT [Veličković et al., 2018]).

Methodology We evaluate depth through layer-wise inspection of hidden node embeddings, according to the protocol proposed in the previous section. Higher accuracy in Logistic Regression classification means that the clusters are more linearly separable. Representations with low similarity should lead to a CKA close to 0 and highly similar representations to a CKA of up to 1. Figure 4.5 illustrates the results of this protocol for three different deep GNN models (16/64-layers) on three benchmark datasets. All results can be reproduced using our openly available repository, which includes all implementations (as disclosed in Appendix B.2).

Identification and Categorization of Pathologies

Our depth evaluation protocol shows evidence of *unhealthy behavior for all of the studied deep GNNs*. We categorize such behavior as two different types of pathologies. Figure 4.4 shows an example of healthy learning behavior as opposed to the pathological trends identified in this work. The following paragraphs elaborate on how these pathologies manifest in the different deep GNNs (Figure 4.5).

Type I Pathology One of the major inefficiencies that we can observe in the examples of Figure 4.5 is a succession of noisy layers. This manifests as a *sequence of layers that show no specific pattern in terms of similarity* (visually producing a noisy square in the similarity matrix) and whose *hidden representations prove not to be linearly separable* (Logistic Regression accuracy equivalent to random classification). We call this pattern a *Type I pathology*. This pathology is very evident in the early layers of GCN+InitRes and GCNII networks, and gets more and more pronounced as we increase network depth because the number of noisy layers seems to increase proportionally with the overall depth of the network. *Texas* seems to be the benchmark that is more affected by this pathology (only the last 3-4 layers appear to contribute to improve the discriminative power of the network, regardless of its overall depth).

Type II Pathology The last layers of GCN+InitRes and GCNII models for *Cora* and *Pubmed* seem to be able to progressively learn, but we can also verify an occasional redundancy of some consecutive layers. This redundancy is showed by *above-random yet semi-constant accuracy* when it comes to the linear separability of the clusters and *nearly identical node representations* (similarity close to 1). In practice, this means that these layers could potentially be pruned with little expected consequences for the final

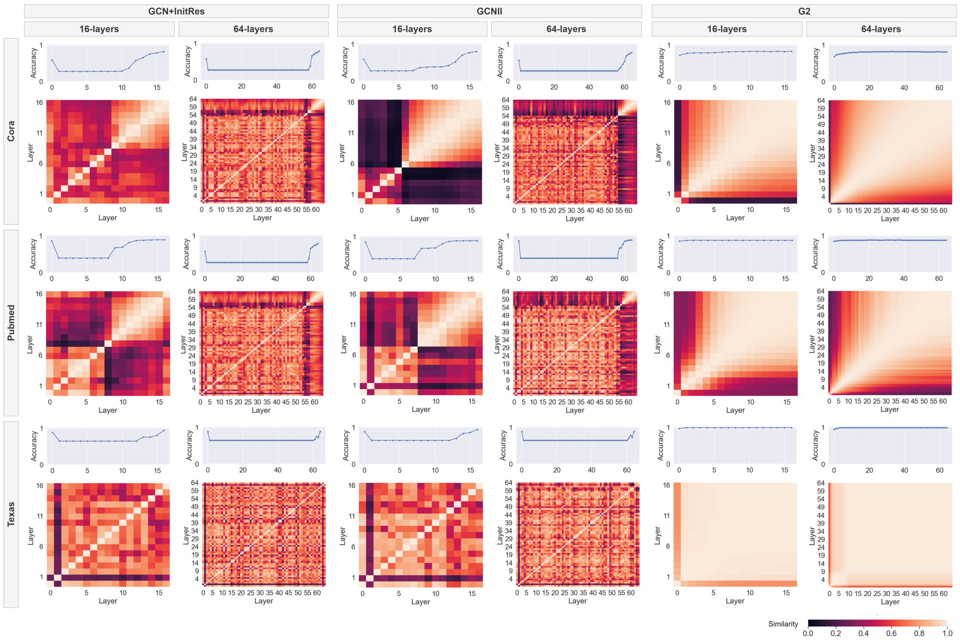


Figure 4.5: Results of our depth evaluation protocol for GCN+InitRes, GCNII and G^2 (16/64-layers) on three benchmarks (Cora, Pubmed, Texas). Layer-wise Logistic Regression is measured in terms of accuracy of classification using the intermediate hidden representations $\mathbf{H}^l \in \{1, ..., L\}$; pairwise CKA of layers $l, k \in \{1, ..., L\}$ is represented by the similarity matrices.

performance of the network. We call this a *Type II pathology*. The impact of this pathology in the overall outcome is very limited for the example described above (as we only observe it for a small number of consecutive layers), but the same pathology can be easily identified for large parts of the G^2 networks. The extent to which we observe this pathology in G^2 models is particularly cumbersome for *Texas*, for which the large majority of every network shows identical node representations.

Severe cases The severity of these pathologies varies with benchmark, model type and depth, but the phenomena are the same. Both types of pathologies can coexist in the same network, with different levels of severity (e.g., 16-layer GCN+InitRes on *Pubmed*). The fact that *Texas* exhibited the most exacerbated examples of the described pathologies is very significant, as *Texas* is a benchmark for which a feature-only MLP remains relevant by delivering state-of-the-art results, even when compared with complex GNN architectures (see *Case 3* of Table 4.2). This observation is not a coincidence, and similar patterns that show severe pathological behaviors can be found for the remaining benchmarks of the feature-dominant regime (see examples in Figure B.3 of Appendix B.2).

These findings resonate with the results from Chapter 3—where feature-rich regimes can suppress the value of structural information. This observation reappears here as a form of depth-induced failure mode, suggesting that the architectural mechanisms enabling deep GNNs also permit, under certain conditions, the complete disregard of graph structure, which we will discuss in further detail in the next section. These parallels strengthen the hypothesis that the effectiveness of GNNs is tightly coupled to their capacity to balance feature and structural learning—either in shallow or deep regimes.

Causes

Going back to the comparison of deep GNN layers in Section 2, we can yet find that these inefficient learning behaviors can be a direct and logical consequence of the key ingredients that have been enabling GNNs to go deep.

Type I $\leftrightarrow$ Preservation of Earlier Embeddings Looking into GCN+InitRes and GCNII in Figure 4.5, we see similar patterns of unhealthy learning, mostly through the development of Type I pathologies. These networks are also the ones that use direct connections to the initial node embeddings $\mathbf{H}^0$ in their layers (Table 4.1). We hypothesize that such connections are responsible for enabling Type I pathologies, as the networks are able to recreate the information lost due to the excessive smoothing of the consecutive graph convolutions in later embeddings, by using the initial representations. For this reason, the first p layers are essentially useless/noisy (see drop in accuracy to random classification level from $\mathbf{H}^0$ to $\mathbf{H}^1$ for all deep GCN+InitRes and GCNII experiments), while the last $L - p$ layers seem to be able to progressively learn useful embeddings. By learning to obliterate the usage of its first layers, the networks are able to deliver $\mathbf{H}^L$ with optimal smoothing for the downstream task. The same optimal level of smoothing could, however, be achieved with a shallower network.

Type II $\leftrightarrow$ Weighting Graph Convolution Pathologies of Type II seem to be related with the coefficients that weigh the graph convolution term and the residual connection. All studied models (GCN+InitRes, GCNII, G^2) consider these coefficients to be symmetrical. The main difference between these approaches is that G^2 learns a scalar per layer (τ^l), while GCN+InitRes and GCNII consider a tunable hyperparameter (α). The results of Figure 4.5 show that G^2 is particularly affected by Type II pathologies, especially in the final layers of the networks. The high similarity of consecutive node embeddings for the deeper representations suggests that, as depth increases, τ^l tends to zero, at which point $\mathbf{H}^l = \mathbf{H}^{l-1}$ and layers become redundant. For *Texas*, a dataset for which graph convolutions do not seem to be useful, this can even mean repetitive representations for all $l > 1$. We expect that, for GCN+InitRes and GCNII networks, α is also playing a role in refraining the smoothing speed, as occasional repetitive representations can be found for the last $L - p$ layers that progressively learn (e.g., semi-constant accuracy and high similarity of 16-layer GCNII on Cora for $8 < l < 11$).

4 Discussion

Following our in-depth analyses, let us further consolidate the insights from Sections 2 and 3 by elaborating on two central findings of this study:

1. Pathologies are a consequence of the addition of residual connections and the smoothing refraining coefficients to GNN layers;
2. All surveyed methods in Table 4.1 rely on both of these *ingredients* to create deep architectures.

The direct correspondence between the causes of the pathologies and the identified key ingredients of depth scaling suggests that similar behavior should be expected whenever these components are part of deep GNN architectures—i.e., our conclusions are not limited to the examples we inspected. For this reason, we expand on the generic relation between the *flat curve* and the pathologies (4.1), on the role of residual connections—one of the most frequently adopted and unquestioned methods in the literature—in GNN learning (4.2), and on how pathologies can not only justify the *flat curve* but also other puzzling phenomena whose causes still lack general agreement in the community, such as the equivalence between GNNs and MLPs on some heterophilic benchmarks (4.3). Finally, we also outline the main limitations of our study and propose future research directions to further investigate and potentially overcome the challenges identified in the development of deep GNN architectures (4.4).

4.1 Performance stagnation over depth: the *flat curve*

The inefficient learning behaviors identified as pathologies in the previous section provide a deeper understanding of the results presented in Table 4.2 for GCN+InitRes and, consequently, the *flat curve* phenomenon.

GCN+InitRes case study *Case 1* benchmarks correspond to datasets for which the underlying structure of the graph encodes relevant information under the smoothing assumptions of graph convolutions (convolution-sufficient regime). These benchmarks benefit from at least one graph convolution operation, which is why vanilla-GCN can perform on par with GCN+InitRes. In these cases, GCN+InitRes develops a pathology of Type I—which ultimately turns a large part of the networks into useless representations—and, occasionally, some cases of redundant representations (Type II); as such, these networks only turn out to effectively take advantage of a small number of convolutions, leading to results equivalent to those of a shallow vanilla-GCN. We hypothesize that this behavior is possible due to the initial residual connection, which enables direct flows of information from the first, non-smoothed levels into the deeper, potentially oversmoothed ones. Analogously, for benchmarks of *Case 2*, we observe pathologies of both types. In this case, GCN+InitRes derives more meaningful representations than its baseline counterparts (complementary regime); however, a shallow GCN+InitRes of 1 or 2 layers is sufficient to achieve top performance. As such, stacking more layers leads to inefficient learning behaviors.

Finally, *Case 3* benchmarks appear inherently ill-suited for benefiting from graph convolutions, as evidenced by the fact that both a feature-only MLP and a GCN+InitRes can achieve top-tier performance on these datasets (feature-dominant regime). The severe pathologies observed in deep networks trained on these benchmarks—whether of Type I or Type II—do not necessarily reflect a failure of the network, but rather an adaptation to the underlying information landscape. As discussed in Chapter 3, datasets with highly informative node features and weak or noisy structure tend to favor architectures that preserve or emphasize feature information, often rendering structural signals redundant or even detrimental. In such settings, pathologies that suppress the effect of graph convolutions (e.g., through feature-preserving residual connections or diminishing convolution weights) serve a functional purpose: they allow the network to sidestep unhelpful structure and default to a form of MLP-like behavior.

This leads to a consistent empirical equivalence between GNNs and MLPs on these benchmarks, with GNNs effectively reducing to stacks of non-linear feature transformations made possible by the initial residual connection or the diminished weighting of the convolution terms. In other words, the very mechanisms designed to enable deep GNNs, when operating in structure-irrelevant regimes, end up erasing the role of the graph, mirroring the findings from Chapter 3 where strong features can mask the benefits of structural learning. This highlights the broader implication that GNN architectures lacking adaptive mechanisms may default to underutilizing graph information in contexts where features dominate, reinforcing the need for models that can dynamically assess and leverage the relative utility of structure and features.

Other deep GNNs These observations for GCN+InitRes could be generalized for the remaining, more complex deep GNNs, since they are sustained by the theoretical analysis of the layers' expressions that led us to conclude that they share some key ingredients. The fact that all of them combine terms that preserve previous, non-smoothed representations and have coefficients that can refrain the smoothing (see Section 2) strongly hints at the conclusion that these networks are undergoing learning handicaps analogous to the ones identified in this work as pathologies when they attempt to go deep. This would justify why we also verify the *flat curve* in their case, and is corroborated by the results exhibited in Figure 4.5 for GCNII and G^2.

Showing exhaustive evidence of the pathologies for the remaining models is out of the scope of this work, but we encourage other researchers to do so. In particular, looking into the layers of GPRGNN and GGCN, we have reasons to believe that analogous pathologies can be happening in their case (recall the layer expressions of these networks in Table 4.1). For example, the ablation studies on γ_l published in the original GPRGNN publication [Chien et al., 2020] show that this variable frequently tends to zero as depth increases (meaning that the deeper, more smoothed representations will be negligible for attaining the final representations; thus, we should expect hidden embeddings of high similarity as we go deeper, making them redundant—Type II pathology). For the GGCN case, the complexity of the layers can hamper its close analysis; in order to be thorough, it would be important to look into all variables (learned or tuned) and study their downstream impact. However, just by looking into η, we find that this function easily tends to zero after a small number of iterations (which can be tuned), again making the smoothing term nearly irrelevant as we go deeper. This rationale should be further investigated through the application of dedicated protocols to provide confirmatory evidence.

Beyond Depth Since the pathologies are rooted in the addition of residual connections and smoothing controlling coefficients to each layer, and not in the convolution operation itself (which can be more or less expressive in some cases), we expect that our conclusions can extend to other architectures (see an example for GAT+InitRes in Figure B.5 of Appendix B.2). Evidently, we do not advocate that the proposition of the new and more complex layers is useless. These models have proved their potential to deliver state-of-the-art results in several benchmarks, which validates their practical utility and relevance within the field of GNN research. We do, however, emphasize that, at this point, we could find no evidence of any benefit of practical utility when scaling them indefinitely with respect to depth, both in the literature and in our own experiments. This finding makes us question whether the pursuit of depth within the field is being properly addressed, as current approaches seem to overshadow the problems with deep GNNs rather then solving them. For this reason, we encourage other researchers to evaluate their claims of appropriate depth scaling more thoroughly, and propose a protocol to assist them in doing so.

4.2 The role of residual connections

The importance of residual connections has been highlighted in previous works [Giovanni et al., 2023; Jaiswal et al., 2022]. However, we clarify the extent of their contribution from an empirical perspective (explicitly for initial residuals, but with implicit assumptions for other architectures), by looking into the intermediate hidden node representations. By doing this, we bring forward, for the first time, the fact that top performing GNNs with residual connections can converge to a solution in which they avoid oversmoothing by reconstructing non-smoothed representations from earlier embeddings in later layers and disregard the noisy information of the previous ones (Type I pathology). This means that while residual connections can make GNNs more powerful to some level, by enabling the combination of smoothed with sharp representations, they are currently not effective as a means to achieve depth.

We verify this for GCN+InitRes, but we cannot rule out that analogous behavior might be observed for more complex architectures, as all the surveyed architectures include coefficients that enable them to preserve previous node representations for indefinite depth (Table 4.1). For this reason, we encourage other researchers to perform similar analyses as the ones performed in this work when proposing new layer types that enable deep architectures by pertaining residual connections (or analogous strategies of preservation of the information of early hidden representations), and not just to evaluate if oversmoothing is avoided using metrics that do not assess their progressive discriminative power, as discussed in 3.1).

4.3 Overcoming harmful heterophily

Several techniques originally proposed to address oversmoothing have also proven empirically effective in heterophilous settings. This has led to a growing body of work exploring the conceptual ties between oversmoothing and heterophily [Yan et al., 2022; Bodnar et al., 2022; Giovanni et al., 2023], often framed through insufficient analytical tools. Our findings add a novel and compelling explanation to this observed overlap: the same architectural mechanisms that enable deep GNNs to preserve early information and mitigate oversmoothing also work as a fallback when graph structure proves uninformative—whether due to depth or to the nature of the structure itself.

In Chapter 3, we saw that while some heterophilous graphs encode rich, class-relevant structure that GNNs can exploit, others present patterns in which local averaging actively obscures discriminative features. In these cases, smoothing operations like those in standard GCNs dilute discriminative signals and hinder learning. Thus, more broadly, we can refer to harmful heterophily as a condensed way of expressing structural configurations that fail to encode task-relevant correlations—regardless of their measured homophily. What we demonstrate in this chapter is that, in such settings, even the most sophisticated GNN architectures—those designed specifically to go deeper and attempt to leverage graph structure more effectively (by theoretically encoding more long-range dependencies)—learn to bypass the graph altogether. Our analysis shows that deep networks exhibiting either Type I or Type II pathologies often revert to using earlier, less-smoothed node representations (or skip message-passing entirely) when graph structure fails to provide added value. This results in the emergence of *MLP-like behavior*, not as a failure of optimization, but as a rational learning strategy. These networks, though structurally equipped to exploit depth and neighborhood information, learn that optimal performance arises from operating primarily in the feature domain—effectively reducing to an MLP wrapped in residuals and nonlinearities.

This provides a new, mechanistic explanation for the persistent empirical finding that GNNs sometimes perform on par with feature-only MLPs on benchmarks such as *Texas* or *Cornell*. Rather than being an artifact of shallow models or poor training regimes, this behavior arises from deeper architectural principles: when structure cannot be productively encoded, the network learns to ignore it. Crucially, we show that this behavior holds even for GNNs with powerful residual mechanisms, smoothed aggregation controls, and layer-wise weighting schemes—architectures previously assumed to be more immune to such degeneracies. To our knowledge, this is the first work to formally demonstrate that these failures are not merely surface-level artifacts but emerge from systematic interactions between architecture, depth, and structure informativeness.

Ultimately, this perspective reframes the challenge of harmful heterophily. It is not enough to merely "handle" heterophily through smoothing control. Instead, we need adaptive architectures capable of recognizing when structural information is useful and when it is not—and adjusting their reliance on it accordingly. This finding opens new avenues for designing GNNs that can selectively engage with the graph, rather than blindly smoothing across it, and that can avoid degenerating into MLP-like behavior in structure-poor regimes while still retaining their full expressive power when structure matters.

4.4 Looking Ahead

While the proposed protocol provides a valuable framework for assessing the learning behavior of deep GNNs, it also presents certain constraints that must be acknowledged. At the same time, we believe this work opens several avenues for future investigations into the design and evaluation of more effective deep GNN architectures.

Limitations Although our protocol addresses critical shortcomings of prior depth evaluation methods, it is primarily designed to support the diagnosis and interpretation of learning inefficiencies, rather than to offer a conclusive single statistical measure for the comparative evaluation of different GNN architectures. This focus, while providing in-depth insights, may restrict the applicability of our method in settings that demand more direct, standardized benchmarking metrics. In fact, given the inherent complexity of GNNs and their capacity to integrate multiple levels of structural information, we argue that no single metric can adequately capture the richness of their behavior without reproducing the limitations observed in earlier approaches. Additionally, this study prioritizes the analysis of how GNNs learn task-relevant representations over the pursuit of state-of-the-art performance. Accordingly, we do not engage in extensive hyperparameter tuning or the exploration of advanced regularization strategies, instead opting for a uniform training setup across all models to ensure fair comparison and a tractable number of replicable experiments. Full implementation details and parameter choices are provided in Appendix A.5 to facilitate reproducibility.

Future Work Building upon the insights gained in this study, future research should aim to validate and refine the hypotheses concerning the origin of the observed pathologies. A key direction will be to investigate whether such pathologies can be mitigated or circumvented through the design of novel architectures or training schemes, potentially enabling more effective exploitation of depth in GNNs. Further development of evaluation tools that balance empirical interpretability with standardizability could also enhance the broader applicability of our approach. Ultimately, we envision that a deeper understanding of GNN learning behavior will inform the design of more robust and scalable models, capable of fully leveraging hierarchical representations without succumbing to learning inefficiencies.

5 Conclusion

The convergence of GNNs and general architectural advances in deep learning methods has led to significant progress in representation learning of structured data. Yet, this work challenges the prevailing assumption that increasing depth inherently yields more powerful models. Through a systematic empirical investigation, we expose the phenomenon of performance stagnation over depth—the *flat curve*—across a spectrum of GNN architectures, from foundational models to state-of-the-art variants. Despite their complexity and depth, these networks frequently fail to outperform their shallower counterparts, with deeper layers often exhibiting noisy or redundant behavior.

This work reframes the prevailing narrative: the drive for depth in GNNs must be met with caution, scrutiny, and rigorous empirical validation. Our findings suggest that certain architectural choices—such as residual connections and smoothing control coefficients—while intended to support depth, can induce learning inefficiencies that undermine their intended purpose. These insights call for a reassessment of widely adopted design patterns in deep GNN research. We advocate for a more critical and principled approach to architectural development, grounded in both empirical clarity and theoretical soundness.

To support this shift, we introduce a protocol for evaluating depth-related performance in GNNs, offering a framework for diagnosing learning pathologies and interpreting hidden representations. As the field matures, we urge researchers to not only demonstrate that their models can go deep, but to provide concrete evidence that such depth confers genuine representational or performance advantages. In doing so, the community can move beyond the superficial scaling of models and toward a more meaningful, task-aligned design paradigm—ensuring that depth in GNNs is not just possible, but purposeful.

5 Informed GNN Design

The insights developed across the previous chapters on feature-structure interplay, depth-related pathologies, and task-specific architectural trade-offs are intimately related and can be combined to bring a unified perspective on GNN development. The aim is not to propose yet another model, but to translate the accumulated evidence into a practical framework for informed GNN design—one that helps researchers and practitioners align architectural choices with the available information and relational properties of their data. By consolidating these insights into concrete guidelines, this chapter moves from individual findings to an integrative, empirically grounded approach for tackling graph learning problems.

Collectively, these insights motivate the formulation of a *principled framework aimed at guiding informed GNN design*, particularly when addressing emerging challenges in graph representation learning. To illustrate its practical utility, a proof-of-concept case study is included in Section 2, focusing on molecular representation learning—a domain of high real-world relevance and complexity. Adopting a tutorial-like perspective, the case study demonstrates how the proposed framework can be operationalized in practice, applying the empirical mindset and architectural insights developed throughout this thesis to address the challenges posed by learning on molecular graphs. This serves not only as a validation of the framework but also as a step-by-step guide for its application in complex settings.

1 Design Framework

The following design framework distills the findings of this thesis into a practical, step-by-step process, providing clear guidance for applying these insights to real-world problems. By following its stages, researchers and practitioners can adopt a more methodic, task-aware mindset that can assist them in extracting the most relevant patterns from graph data using GNNs.

(Informed) GNN Design Framework

- **Enforce Geometric Priors in Intra-Layer Design**
  - Encode relevant geometric priors (graph- and/or domain-specific) in intra-layer design, as symmetry-preserving layers are a fundamental theoretical and practical pillar of GNN learning—allowing them to narrow down the (otherwise unfeasible) space of functions that they can approximate (Chapter 2).
- **Validate the Role of Graph Structure**
  - Assess whether the graph structure provides meaningful information that is relevant to the task under GNNs' smoothing assumptions (e.g., by using feature-structure decoupling methods); otherwise, simpler, structure-agnostic methods might be sufficient or even outperform them (Chapter 3).
- **Exploit Beneficial Smoothing**
  - Use simple graph convolutions when local smoothing aligns with class boundaries, as they effectively filter noisy features in semi-supervised node tasks. These cases offer the greatest benefit, where complex GNNs—often defaulting to feature preservation—tend to underperform (Chapter 3).
- **Limit the Depth of GNNs**
  - Be conservative about the number of layers: use as many layers as needed, but as few as necessary. Most benchmarks show no benefit from deep architectures. Deeper networks risk performance plateaus and inefficient learning (Chapter 4).
- **Consider Residual Connections to Improve Learning (not to Go Deeper)**
  - Use residual connections with overall awareness that they can enhance learning and create more discriminative embeddings for some tasks; however, keep in mind that they do not guarantee effective depth utilization (Chapter 4).
- **Diagnose Learning Dynamics and Bottlenecks**
  - Go beyond oversmoothing metrics, avoiding ill-fundamented diagnoses of learning bottlenecks. Analyze intermediate embeddings for signs of degeneration—such as overly similar or noisy representations—to guide architectural choices (Chapter 4).

2 Applying Informed GNN Design Principles to Molecular Generation Applications: a Case Study

In this case study, we investigate the application of GNNs in molecular representation learning through the lenses of our empirical-driven mindset and the comprehensive insights on GNN architecture design obtained throughout this thesis.

Molecular representation learning consists of an increasingly relevant domain in computational chemistry, drug discovery, and materials science, and it is currently one of the most research-intensive areas of application of graph-based learning. Particularly, generative approaches have gained traction in recent years following the success of diffusion models and their potential for in-silico drug design. This is the precise reason why we selected this domain, as it will enable us to *test our framework in a highly challenging and competitive scenario.* We argue that a deep understanding of the inductive biases and benefits and limitations of GNNs—particularly as applied to graphs with strong domain-specific constraints—can significantly enhance their effectiveness in this domain.

Domain-Specific Constraints of Molecular Graphs

Molecular graphs present a unique set of topological and semantic constraints. Each molecule is modeled as a graph where nodes correspond to atoms and edges to chemical bonds. These graphs are inherently sparse, with well-defined local connectivity patterns and a strong dependence on domain-specific priors which can even include chemistry and physics constraints. Furthermore, many molecular properties are functions not just of connectivity but of 3D conformation, which are only indirectly accessible from the 2D graph representation. As such, molecules serve as a concrete and practical instantiation of the abstract graph concept, where the inclusion of 3D atomic coordinates enriches the purely combinatorial structure with continuous spatial information.

This explicit embedding of geometry introduces Euclidean dynamics into the fundamentally non-Euclidean graph domain, necessitating models that can jointly reason over both relational topology and spatial arrangement. These properties make molecular graphs a highly structured and information-rich domain for graph learning, but they also impose constraints on model design. In particular, GNNs must respect key invariances (e.g., permutation and rotation invariance) and capture both local and global chemical context.

Motivation for Informed GNN Design in Molecular Generation Applications

While molecular representation learning has historically focused on applications like property prediction tasks, recent advances have shifted attention towards generative modeling, particularly through diffusion-based approaches [Hoogeboom et al., 2022; Xu et al., 2023]. Molecular generation poses unique challenges for GNNs, requiring the capacity to model complex distributions over valid molecular graphs while preserving domain-specific constraints such as chemical validity, geometric consistency, and diversity. Models like the Equivariant Graph Neural Network (EGNN) [Satorras et al., 2021] have shown strong performance in molecular generation tasks, particularly when integrated into diffusion-based frameworks [Hoogeboom et al., 2022]. We posit that systematically analyzing such architectures through our informed GNN design framework offers a valuable opportunity to understand the learning dynamics at play, assess design adequacy, and diagnose potential bottlenecks.

In this case study, we demonstrate how applying our framework to a high impact real-world challenge can uncover insights into architectural behavior—such as smoothing effects, depth efficiency, and inductive bias alignment—and inform principled modifications that further enhance performance or improve learning dynamics.

2.1 Experimental Setup

To demonstrate the practical value of our proposed framework, our case study analyzes the EGNN architecture [Satorras et al., 2021]. Our goal was to assess a state-of-the-art GNN, which explicitly incorporates domain-specific geometric inductive biases, under the lens of our systematic guidelines. This exercise serves as a proof-of-concept, validating the applicability of our framework in diagnosing model behavior, informing design decisions, and enhancing practical performance in molecular representation learning tasks.

In particular, we focus on the EGNN architecture, as originally introduced by Satorras et al. [2021], and specifically as instantiated within the E(3)-Equivariant Diffusion Model (EDM) framework proposed by Hoogeboom et al. [2022] for molecular generation. The EGNN is a message-passing neural network designed to preserve equivariance to Euclidean transformations, ensuring that its learned representations remain consistent under translations, rotations, and reflections of molecular geometries. Under EDM, the EGNN model simultaneously processes continuous features, such as atomic coordinates, alongside categorical features like atom types, and is trained to denoise a diffusion process. This means that the EDM model leverages the EGNN within a generative diffusion process, where molecular structures are progressively denoised from a noisy prior distribution back to chemically valid configurations. This approach has demonstrated strong performance in generating molecules with realistic structural and chemical properties, highlighting the practical relevance of analyzing EGNN in this generative context.

We selected the QM9 dataset for this analysis, which contains approximately 130,000 small organic molecules annotated with a range of quantum chemical properties. QM9 is among the most widely used benchmarks for molecular representation learning and generative modeling due to its standardized format, chemical diversity within small molecules, and broad adoption in the literature. Moreover, the accessibility and well-characterized nature of QM9 make it well-suited for a tutorial-like case study, allowing us to systematically demonstrate the application of our diagnostic framework in a controlled yet practically meaningful setting.

This experimental setup enables us to critically examine the EGNN within a state-of-the-art generative modeling pipeline, assess its architectural adequacy, and explore the possibility of targeted interventions guided by our framework.

2.2 Analysis and Enhancement of EDM for Molecular Generation

1) Ensuring Geometric Inductive Biases

The first diagnostic step in our framework involves verifying that the chosen GNN architecture embeds the necessary inductive biases aligned with the problem domain. For molecular learning, this specifically requires equivariance to rotations, translations, and invariance to permutations—properties essential for accurately capturing molecular geometry and symmetries. The EGNN satisfies these criteria through an intra-layer design that couples updates of node features and coordinates while preserving the relevant symmetries.

Formally, the EGNN updates node features and positions jointly via:

$$\mathbf{h}_u^{(l+1)} = \phi_h(\mathbf{h}_u^{(l)}, \sum_{v \neq u} \tilde{e}_{uv}\mathbf{m}_{uv}) \quad ,$$

$$\mathbf{x}_u^{(l+1)} = \mathbf{x}_u^{(l)} + \sum_{v \neq u} \frac{\mathbf{x}_u^{(l)} - \mathbf{x}_v^{(l)}}{d_{uv} + 1} \phi_x(\mathbf{h}_u^{(l)}, \mathbf{h}_v^{(l)}, d_{uv}^2, a_{uv}) \quad ,$$

$$\text{where} \quad \mathbf{m}_{uv} = \phi_m(\mathbf{h}_u^{(l)}, \mathbf{h}_v^{(l)}, d_{uv}^2, a_{uv}) \, , \; \tilde{e}_{uv} = \phi_e(\mathbf{m}_{uv}) \text{ and } d_{uv} = \|\mathbf{x}_u^{(l)} - \mathbf{x}_v^{(l)}\|_2 \, .$$

This formulation enables the model to update node features $\mathbf{h}_u$ based on both relational attributes and interatomic distances, while updating coordinates $\mathbf{x}_u$ through distance-based adjustments that preserve Euclidean equivariance. The combination of feature and coordinate updates mediated by the learned functions ϕ_h, ϕ_x, ϕ_m, and ϕ_e ensures that both the topology and geometry of the molecular graph are coherently integrated into the representation. This principled layer design confirms that EGNN is theoretically well-grounded for molecular graph learning, where spatial configuration is not only informative but often decisive for property prediction and generative tasks.

Table 5.1: Performance comparison on QM9 property prediction targets for EGNNs of different depths and a structure-agnostic baseline (Mean Absolute Error—lower is better). Benchmark results for the structure-agnostic were extracted from Faber et al. [2017], according to the comparison method of Gilmer et al. [2017]. Results for 7-layers EGNN (EGNN-7) were directly extracted from Satorras et al. [2021]. Results for EGNN of 2 and 12 layers (EGNN-2 and EGNN-12, respectively) were computed using the code released by Hoogeboom et al. [2022] for 1000 epochs.

Property	α	$\Delta\epsilon$	ϵ_{HOMO}	ϵ_{LUMO}	μ	C_v
Unit	bohr^3	eV	eV	eV	D	cal/mol K
Structure-agnostic*	0.175	0.107	0.066	0.084	0.334	0.044
EGNN-2	0.125	0.086	0.056	0.052	0.071	0.046
EGNN-7†	**0.071**	**0.048**	**0.029**	**0.025**	**0.029**	**0.031**
EGNN-12	0.093	0.063	0.039	0.034	0.048	0.039

* Best non-message passing method in Faber et al. [2017].
† According to Satorras et al. [2021] (original EGNN paper).

2) Verifying Structure Informativeness

The second diagnostic step in our framework is to empirically verify whether the structural information encoded by the GNN is effectively leveraged to improve task performance. For this, we analyzed EGNN performance on the QM9 property prediction task, comparing it against a non-message passing baseline that does not exploit relational structure explicitly [Faber et al., 2017]. We selected the best performing method using traditional machine learning models for a challenging comparison, following the method of [Gilmer et al., 2017], and consider this our structure-agnostic baseline. As shown in Table 5.1, EGNN markedly outperforms the structure-agnostic method across all evaluated quantum chemical properties. This confirms that the network is indeed capturing and exploiting graph connectivity and geometric information to produce more discriminative representations.

Complementing this, results from the generative setting (Table 5.2) further corroborate this finding: the EDM model, which employs EGNN as the core diffusion backbone, achieves significantly better performance than the models that preceded it in molecule generation on QM9 (ENF and G-Schnet), notably improving on stability, validity, and uniqueness. While structure-agnostic baselines are impractical to reproduce in the diffusion setting, the combined evidence from property prediction and generative modeling shows that EGNN productively encodes structural information, with consistent empirical gains that validate its theoretical foundations.

3) Evaluating the Smoothing Regime

Another critical guideline in our framework is assessing whether the employed graph convolutions (i.e., aggregation through message-passing) lead to beneficial feature smoothing relative to the task. While our prior analyses focused on semi-supervised node classification tasks, this case study shifts focus to whole-graph representations, but we expect that equivalent analyses can be performed in this context. According to what we expect in a regime of beneficial smoothing, the study in Table 5.1 evidences the sustained outperformance of EGNN over structure-agnostic baselines, indicating that message-passing successfully produces task-relevant signals through feature mixing, ideally reinforcing meaningful similarities while filtering out noise across layers. Notably, the EGNN does not incorporate mechanisms to preserve or reintegrate early-layer representations—such as residual connections or explicit feature preservation pathways—into successive hidden node states; this also supports that no explicit masking of potential harmful smoothing is encoded in EGNN layers. Given this context, while a dedicated study of the smoothing regime of EGNN under the EDM framework is computationally impractical, with prohibitive demands, we expect that this smaller scale experiment on property prediction, along with the analysis of the layer elements, can be a good indication of an appropriate smoothing regime for the generative application as well. The next paragraph continues this discussion, also relating it with the principle of depth tuning.

4) Tuning Depth

Consistent with our design principles, we examined how EGNN's depth was configured in the molecular generation setting of EDM. While the original publications do not explicitly report depth tuning, our inspection of the released code shows that the number of layers is exposed as a configurable hyperparameter during training and evaluation, suggesting that depth was treated as a tunable parameter in practice. Although we did not re-run the EDM pipeline for different depths due to its computational complexity, insights from our property prediction experiments on QM9 provide a strong basis for extrapolation. In particular, we see that the authors also exposed the number of layers as a tunable hyperparameter and the experiment in Table 5.1 corroborates that they used this strategy to find an optimal depth of 7 layers (EGNN-7 achieves superior performance relative to both shallower, EGNN-2, and deeper, EGNN-12, variants). Importantly, this strong performance is achieved without relying on architectural mechanisms explicitly designed to enable arbitrarily deep networks, such as smoothing control coefficients or feature preservation strategies. This is the case for both the property prediction application and the generative application alike.

While a thorough direct comparison of different depths within EDM remains computationally challenging, it is reasonable to assume that a similar depth tuning

Table 5.2: Performance comparison on QM9-based molecule generation. The metrics assess atom- and molecule-level stability, chemical validity, and the proportion of valid and unique samples. Our enhanced EDM model, which integrates initial residual connections into EGNN, achieves state-of-the-art results. Benchmark results were extracted from Xu et al. [2023].

	Atom Stability (%)	**Molecule Stability (%)**	**Validity (%)**	**Valid & Unique (%)**
Data	99.0	95.2	97.7	97.7
ENF	85.0	4.9	40.2	39.4
G-Schnet	95.7	68.1	85.5	80.3
GDM	97.0	63.2	-	-
GDM-AUG	97.6	71.6	90.4	89.5
EDM	98.7	82.0	91.9	90.7
EDM-Bridge	98.8	84.6	92.0	90.7
GraphLDM	97.2	70.5	83.6	82.7
GraphLDM-AUG	97.9	78.7	90.5	89.5
GeoLDM	98.9	89.4	93.8	92.7
EDM-*Enhanced* (ours)	**99.1**	**90.2**	**95.8**	**95.7**

process was applied in the EDM setting (which shares a code base and was performed by the same authors), achieving an EGNN configuration that ultimately reflects an empirically selected balance between expressivity and stability. The attained depth level can, thus, exemplify a well-calibrated smoothing regime that maximizes signal propagation while preserving task-relevant information—consistent with our framework's principle of employing "as many layers as needed, and as few as necessary."

5) Improving Learned Representations with Residual Connections

As established in the previous analyses, the EGNN architecture was configured to operate within a regime of beneficial smoothing, where depth is carefully tuned to optimize the trade-off between signal propagation and information retention, without relying on architectural mechanisms that mask inefficiencies in deeper networks. This reflects a central idea of our framework: calibrate depth to the task rather than pursue it indiscriminately. However, even within this balanced regime, our framework also emphasizes that residual connections can serve as targeted interventions to improve representation learning—not to simply facilitate deeper networks, but to encourage more discriminative and expressive embeddings when beneficial.

Following this principle, we identified an opportunity to enhance the EGNN's capacity for forming task-relevant representations by introducing an initial residual connection, while preserving the model's core equivariant properties. We applied this modification to the EGNN within the EDM pipeline for QM9-based molecule generation, yielding the *EDM-Enhanced* variant (Table 5.2). This simple yet principled change led to notable improvements across all evaluation metrics, including atom and molecule stability, chemical validity, and uniqueness. These gains illustrate the value of our framework for guiding principled, targeted interventions that enhance representational learning in molecular generative models.

3 Conclusion

This case study demonstrates the practical relevance and diagnostic power of our informed GNN design framework. By systematically analyzing the EGNN within the context of molecular generative modeling, we validated the framework's capacity to elucidate the alignment between architectural choices and domain-specific constraints. Each step of the framework guided a structured assessment that ultimately uncovered a simple yet effective enhancement through the introduction of residual connections, which improved the representational power of the network while preserving important geometric priors. The resulting performance gains on molecule generation tasks not only underscore the framework's utility but also exemplify how informed design can translate into tangible improvements in complex applications.

More broadly, this case study reinforces the central thesis of this dissertation: that effective GNN design is best approached through principled co-design of graph formulations and neural architectures, tailored to the relational and feature characteristics of the problem at hand. Rather than relying on the incremental layering of complexity or depth, our paradigm advocates for leveraging domain knowledge and theoretical insights to constrain the architectural search space in a meaningful and empirically grounded manner.

By applying this perspective, we bridge the gap between theoretical understanding and practical deployment. This alignment advances the methodological foundations of GNN research, promoting a more principled scientific culture, essential for sustained progress both within graph learning and in the broader field of representation learning.

6 Conclusion

This dissertation has systematically explored the capabilities and limitations of GNNs, offering a comprehensive evaluation of how their representational power unfolds across diverse graph structures and feature regimes. Given the centrality of graphs in representing relational data across numerous real-world applications, understanding how GNNs interact with the structure and features of input data is paramount. Graphs serve as a powerful abstraction across domains, but their irregular structure and the high variability in how they encode task-relevant information make effective GNN design nontrivial, motivating the need for more principled approaches that unequivocally inform us on what is needed for these networks to learn adequate representations over graph structures.

Although GNNs have shown promise in capturing complex relational patterns across a wide range of domains, their effective design and application remain fraught with unresolved challenges. Key limitations such as oversmoothing—where deeper models cause node representations to become indistinguishable—, oversquashing—where information from distant nodes is compressed and lost—, and heterophily—where nodes of different classes are more connected than same-class nodes—are frequently considered to undermine performance, posing difficulties that standard GNN architectures often fail to address robustly. However, it is difficult to understand what exactly is the absolute cause of performance degradation as these and other effects are often entangled and can be misinterpreted, leading to misleading conclusions regarding their effective impact in performance degradation.

Beyond these inherent modeling issues, the field suffers from a lack of consistent performance across benchmarks and architectures. Many proposed methods show gains on specific datasets or tasks but fail to generalize, leading to contradictory results and a fragmented understanding of when and why GNNs succeed or fail. This inconsistency is exacerbated by insufficiently rigorous or standardized empirical analyses: evaluation metrics often show significant limitations, comparisons omit important baselines, and experimental designs rarely disentangle the complex interplay between graph topology, feature information, and architectural choices. As a result, many conclusions remain tentative or overly optimistic, hindering the community's ability to draw actionable insights and to develop truly robust, generalizable models.

Recognizing these gaps, this thesis aimed to provide a principled, data-driven framework to better understand the conditions under which GNNs perform well, and to guide architectural decisions accordingly. By combining theoretical rigor with comprehensive empirical evaluation—disentangling the roles of graph structure and node features, assessing the impact of depth and complexity, and diagnosing failure modes—this work contributes to establishing more reliable, interpretable, and efficient GNN designs. Ultimately, it seeks to move the field towards more principled and empirically-sound foundations built on reproducible evidence and graph-specific awareness.

Bridging Theory and Empirical Practice: A New Perspective on GNN Design

To pursue this goal, we structured the thesis around three core investigations.

Chapter 2 positioned GNN research at the intersection of deep learning and geometric reasoning. It laid the main theoretical foundations for graph representation learning, highlighting how the irregular and high-dimensional nature of graph data affects architectural decisions. Furthermore, the chapter argued that the space of possible GNN designs is not only vast but also heavily contingent on properties of the input graph—such as topology and feature distribution—making principled guidance essential.

Building on this foundation, *Chapter 3* introduced a diagnostic framework for disentangling the influence of graph structure and node features on learning outcomes. Through a combination of synthetic and real-world benchmarks, this analysis revealed that GNN performance is highly sensitive to the relative strength and alignment of structural and feature-based signals. These findings offered concrete guidelines for selecting architectures based on the dominant characteristics of a given task.

Finally, *Chapter 4* revisited the widely held assumption that increasing depth improves model expressivity. Through a new evaluation protocol designed specifically for GNNs, we demonstrated that deeper architectures often lead to redundancy, noisy representations, and ultimately underuse of structural information—sometimes collapsing into behavior akin to MLPs. This analysis provided both theoretical clarity and practical tools for evaluating depth in GNNs, reinforcing the value of shallow, well-matched architectures.

Taken together, these contributions establish a coherent empirical paradigm for GNN research, brought forward in *Chapter 5*: one that moves beyond general-purpose complexity and instead promotes data-aware, problem-driven design. By combining theoretical insight with methodological rigor, this thesis advances our understanding of when, why, and how GNNs should be deployed for effective representation learning.

From Questions to Criteria: What Makes GNNs Perform and Learn Efficiently

This thesis was guided by two central questions: *What are the key requirements for the effective performance of GNNs?* and *Which design choices enable GNNs to learn better representations, more efficiently?*

In response to the first, we demonstrated that performance depends critically on the alignment between data characteristics and architectural assumptions. GNNs excel when the relational structure offers a useful inductive bias that complements or compensates for limited node feature information. This is particularly evident in cases where local connectivity aligns with label distribution—a scenario that GCNs are well-equipped to exploit. Conversely, when structure is not informative for a given task, GNNs often underperform or regress to behavior similar to structure-agnostic models. This failure is not merely due to oversmoothing or heterophily, as previous works frequently oversimplify, but a fundamental mismatch between model biases and graph properties. Using a feature-structure decoupling method, we diagnosed how misaligned graphs lead even advanced GNNs to default to feature-preserving encodings, limiting their effectiveness. These results underscore that structural information is not universally helpful; instead, its utility depends on task-specific alignment.

As for the second question, our findings reveal that design choices such as depth, residual connections, and feature-preserving operations yield benefits only when matched to specific conditions. Shallow architectures often suffice and may even outperform deeper models, especially when structural signals are strong and well-aligned with the learning objective. Moreover, depth increases frequently introduce redundancy and noise, highlighting the importance of principled architectural evaluation over intuition-driven scaling. Thus, the most efficient GNNs are those whose capacity is tailored to the complexity and informativeness of the task environment.

Advancing Empirical Standards in GNN Research

The contributions of this thesis are grounded in rigorous empirical methodologies and critical experimental design, reflecting our commitment to improving empirical research quality within machine learning. Our work was published and divulged at venues that share this vision, such as the *Learning on Graphs Conference* (LoG), the *I Can't Believe It's Not Better* workshop at NeurIPS (ICBINB), and the *Transactions of Machine Learning Research* journal (TMLR), all of which emphasize better empirical standards and more reliable scientific reviews. We see this alignment as not merely a validation of our work, but a meaningful step towards a stronger, more transparent field.

In summary, this thesis advances the understanding of GNNs by:

- Demonstrating a measurable feature-structure interplay that critically impacts GNN performance;
- Defining actionable, empirically-backed guidelines for GNN architecture selection based on task characteristics;
- Introducing a principled framework for evaluating depth and identifying common pathologies in GNNs;
- Contributing to the broader machine learning community through high-quality empirical research practices.

Concretely, these findings advocate for a more careful, problem-driven approach to GNN design and graph problem formulation. In particular,

Task-Aware Architectural Design Our results challenge the default pursuit of deep or complex GNNs, showing that shallow models often perform better at less computational expense. When structure carries most of the useful information, simple GNNs without feature-preserving mechanisms yield more effective representations. Conversely, when node features are rich and informative, architectural elements like residual connections help the network resort to more feature-preserving representations, which can be suboptimal in some cases.

Rethinking Depth and Complexity We find that deep GNNs frequently show evidence of redundant or noisy intermediate hidden representations and fail to leverage additional layers meaningfully. Using the evaluation protocol in Chapter 4, we show that many deep GNNs collapse into MLP-like behavior, undercutting the structural motivations for GNNs. Our evidence supports controlled shallowness, enhanced with appropriate layer configurations, tailored to the downstream task.

Data-Centric Model Selection To support informed architecture choices, we propose an empirical framework that analyzes the informativeness of structure vs. features and detects representational inefficiencies. This approach enables matching GNN capacity and design to the problem's inductive biases, shifting focus from model-centric trial-and-error to principled, data-driven design.

Improving Empirical Standards Our contributions go beyond model proposals: we emphasize reproducible, task-grounded experimentation as essential to meaningful progress in GNN research. Published in venues that align with this view, our work reflects a broader push for reliable, interpretable, and empirically sound graph learning practices.

Future Challenges

Looking ahead, while this thesis provides a principled foundation for understanding and optimizing GNN architectures and graph problem formulations, several challenges remain. As graphs become increasingly large, heterogeneous, and dynamic in real-world applications, the assumptions underpinning current GNN models—such as local similarity and static relational patterns—may not hold universally. Furthermore, the co-design of architectures and problem formulations, although powerful, may become more complex in multi-relational or hierarchical graphs, where task-relevant signals are spread unevenly across different levels of abstraction.

Addressing these challenges will require *extending the empirical mindset promoted in this work: embracing systematic evaluations, task-specific diagnostics, and adaptive designs that respond to the structure and dynamics of data.* By doing so, the graph learning community can move towards building GNNs that are not only more performant but also more adaptable and robust in the face of real-world complexity.

From Fixed Biases to Adaptive GNNs We believe that a key direction for future research lies on moving beyond fixed architectural biases that hard-code assumptions about the importance of structure and features within the convolution operation. Current GNNs often impose static choices—such as fixed levels of structural smoothing or feature preservation—that limit their adaptability across tasks with different signal profiles. Developing more flexible architectures that prove successful in dynamically adjusting these inductive biases during training would be an important line of research. One promising avenue towards this goal could be the development of co-attention mechanisms that jointly reason over the graph's topology and feature content, allowing the model to selectively focus on the most relevant structural patterns and feature dimensions for each task, and not simply computing attention as a scalar weighting the aggregation of neighbors as has been done so far. Unlike standard scalar attention on neighbors, these mechanisms could enable richer, context-aware interactions across both graph structure and feature space. Similarly, adaptive masking or aggregation strategies could modulate how much the model relies on neighborhood information based on its task-specific utility.

Guiding Future Design with Empirical Tools The diagnostic framework developed in this thesis offers a foundation for such adaptive mechanisms. By quantifying the contributions of structure and features to learning performance, future work can identify where adjustments in message-passing are most needed, guiding the design of more flexible GNNs that better align with task demands. In summary, embracing these empirical tools enables a shift from intuition-driven to evidence-based GNN design. This will help create architectures that dynamically balance structural and feature information, efficiently allocate capacity, and adapt more robustly to the complexity of real-world graphs.

Final Remarks

Thus, overall, this dissertation proposes a *new empirical paradigm*: rather than seeking universally deeper or more complex GNNs, practitioners should *co-design graph formulations and network architectures based on a principled understanding of the problem's relational and feature characteristics*. By treating graphs not merely as inputs but as structured hypotheses about task-relevant relations, and by aligning models with these structures through principled design, this approach promises more efficient, theoretically sound, and successful applications of GNNs across domains. By bridging the gap between theoretical insights and empirical validation, we believe these findings pave the way for more robust, efficient, and domain-adapted graph learning systems, fostering progress both within graph machine learning and across the wider landscape of representation learning. This perspective advances not only the technical frontiers of GNN research, but also its methodological foundations, promoting a scientific culture grounded in transparency and empirical rigor.

Bibliography

Agarwal, O., H. Ge, S. Shakeri, and R. Al-Rfou
2021. Knowledge Graph Based Synthetic Corpus Generation for Knowledge-Enhanced Language Model Pre-training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou, eds., Pp. 3554–3565, Online. Association for Computational Linguistics. Cited on page 29.

Akansha, S.
2025. Over-Squashing in Graph Neural Networks: A Comprehensive survey. Cited on page 32.

Alon, U. and E. Yahav
2021. On the Bottleneck of Graph Neural Networks and its Practical Implications. In *International Conference on Learning Representations.* Cited on page 76.

Arnaiz-Rodriguez, A. and F. Errica
2025. Oversmoothing, "Oversquashing", Heterophily, Long-Range, and more: Demystifying Common Beliefs in Graph Machine Learning. Cited on pages 77, 87, and 114.

Ba, J. L., J. R. Kiros, and G. E. Hinton
2016. Layer Normalization. Cited on page 65.

Bahdanau, D., K. Cho, and Y. Bengio
2016. Neural Machine Translation by Jointly Learning to Align and Translate. Cited on page 65.

Balcilar, M., G. Renton, P. Héroux, B. Gaüzère, S. Adam, and P. Honeine
2021. Analyzing the Expressive Power of Graph Neural Networks in a Spectral Perspective. In *International Conference on Learning Representations.* Cited on pages 31, 84, 107, and 114.

Barron, A.
1993. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945. Cited on page 61.

Batalha, L.
2025. ArXiv Trends. https://arxiv-trends.com/. Cited on pages 30 and 171.

Battaglia, P. W., J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu
2018. Relational inductive biases, deep learning, and graph networks. Cited on pages 71 and 78.

Belkin, M. and P. Niyogi
2003. Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural Computation*, 15(6):1373–1396. Cited on page 27.

Bellman, R. and R. Kalaba
1959. A mathematical theory of adaptive control processes. *Proceedings of the National Academy of Sciences*, 45(8):1288–1290. Cited on pages 60 and 78.

Black, M., Z. Wan, A. Nayyeri, and Y. Wang
2023. Understanding oversquashing in GNNs through the lens of effective resistance. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *ICML'23*, Pp. 2528–2547, Honolulu, Hawaii, USA. JMLR.org. Cited on page 76.

Bodnar, C., F. Di Giovanni, B. Chamberlain, P. Lió, and M. Bronstein
2022. Neural Sheaf Diffusion: A Topological Perspective on Heterophily and Oversmoothing in GNNs. In *Advances in Neural Information Processing Systems*, volume 35, Pp. 18527–18541. Curran Associates, Inc. Cited on pages 28, 32, 108, 122, and 131.

Bordes, A., N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko
2013. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc. Cited on page 29.

Borgwardt, K. and H. Kriegel
2005. Shortest-path kernels on graphs. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, Pp. 8 pp.–. Cited on page 46.

Brock, A., S. De, S. L. Smith, and K. Simonyan
2021. High-Performance Large-Scale Image Recognition Without Normalization. In *Proceedings of the 38th International Conference on Machine Learning*, Pp. 1059–1071. PMLR. Cited on page 65.

Brockschmidt, M.
2020. GNN-FiLM: Graph Neural Networks with Feature-wise Linear Modulation. In *Proceedings of the 37th International Conference on Machine Learning*, Pp. 1144–1152. PMLR. Cited on pages 100 and 174.

Brody, S., U. Alon, and E. Yahav
2022. How Attentive are Graph Attention Networks? In *International Conference on Learning Representations*. Cited on page 69.

Bronstein, M. M., J. Bruna, T. Cohen, and P. Veličković
2021. Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges. Cited on pages 34, 54, 57, 61, 62, 64, 69, 70, and 78.

Bruna, J., W. Zaremba, A. Szlam, and Y. LeCun
2014. Spectral Networks and Locally Connected Networks on Graphs. Cited on page 28.

Cai, C. and Y. Wang
2020. A Note on Over-Smoothing for Graph Neural Networks. Cited on pages 84 and 107.

Castellana, D. and F. Errica
2023. Investigating the Interplay between Features and Structures in Graph Learning. Cited on pages 87 and 108.

Chen, M., Z. Wei, Z. Huang, B. Ding, and Y. Li
2020. Simple and Deep Graph Convolutional Networks. In *Proceedings of the 37th International Conference on Machine Learning*, Pp. 1725–1735. PMLR. Cited on pages 28, 32, 69, 74, 108, 113, 116, 117, and 122.

Chiang, W.-L., X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh
2019. Cluster-GCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, Pp. 257–266, New York, NY, USA. Association for Computing Machinery. Cited on page 74.

Chien, E., J. Peng, P. Li, and O. Milenkovic
2020. Adaptive Universal Generalized PageRank Graph Neural Network. In *International Conference on Learning Representations*. Cited on pages 69, 113, 116, 117, and 130.

Chung, F.
1997. *Spectral Graph Theory*, volume 92 of *CBMS Regional Conference Series*. Conference Board of the Mathematical Sciences. Cited on page 48.

Cong, W., M. Ramezani, and M. Mahdavi
2021. On Provable Benefits of Depth in Training Graph Convolutional Networks. In *Advances in Neural Information Processing Systems*, volume 34, Pp. 9936–9949. Curran Associates, Inc. Cited on pages 107, 108, and 114.

Cordonnier, J.-B., A. Loukas, and M. Jaggi
2020. On the Relationship between Self-Attention and Convolutional Layers. In *Eighth International Conference on Learning Representations*. Cited on page 65.

Cybenko, G.
1989. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314. Cited on page 60.

Dauphin, Y. N., R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio
2014. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, volume 2 of *NIPS'14*, Pp. 2933–2941, Cambridge, MA, USA. MIT Press. Cited on page 60.

Davari, M., S. Horoi, A. Natik, G. Lajoie, G. Wolf, and E. Belilovsky
2022. On the Inadequacy of CKA as a Measure of Similarity in Deep Learning. In *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*. Cited on page 123.

Defferrard, M., X. Bresson, and P. Vandergheynst
2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, Pp. 3844–3852, Red Hook, NY, USA. Curran Associates Inc. Cited on page 69.

Derrow-Pinion, A., J. She, D. Wong, O. Lange, T. Hester, L. Perez, M. Nunkesser, S. Lee, X. Guo, B. Wiltshire, P. W. Battaglia, V. Gupta, A. Li, Z. Xu, A. Sanchez-Gonzalez, Y. Li, and P. Velickovic
2021. ETA Prediction with Graph Neural Networks in Google Maps. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, Pp. 3767–3776, Virtual Event Queensland Australia. ACM. Cited on page 29.

Dwivedi, V. P. and X. Bresson
2021. A Generalization of Transformer Networks to Graphs. Cited on page 78.

Elman, J. L.
1990. Finding Structure in Time. *Cognitive Science*, 14(2):179–211. Cited on page 63.

Euler, L.
1741. Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae*, Pp. 128–140. Cited on page 26.

Faber, F. A., L. Hutchison, B. Huang, J. Gilmer, S. S. Schoenholz, G. E. Dahl, O. Vinyals, S. Kearnes, P. F. Riley, and O. A. von Lilienfeld
2017. Machine learning prediction errors better than DFT accuracy. *Journal of Chemical Theory and Computation*, 13(11):5255–5264. Cited on page 140.

Feng, G., H. Wang, and C. Wang
2023. Search for deep graph neural networks. *Information Sciences*, 649:119617. Cited on page 116.

Fey, M. and J. E. Lenssen
2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*. Cited on pages 43, 172, and 174.

Gao, Y., H. Yang, P. Zhang, C. Zhou, and Y. Hu
2020. Graph Neural Architecture Search. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, Pp. 1403–1409, Yokohama, Japan. International Joint Conferences on Artificial Intelligence Organization. Cited on pages 74 and 79.

Gasteiger, J., A. Bojchevski, and S. Günnemann
2018. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *International Conference on Learning Representations*. Cited on pages 100, 113, 116, and 174.

Gilmer, J., S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl
2017. Neural Message Passing for Quantum Chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, Pp. 1263–1272. PMLR. Cited on pages 71, 78, and 140.

Giovanni, F. D., J. Rowbottom, B. P. Chamberlain, T. Markovich, and M. M. Bronstein
2023. Understanding convolution on graphs via energies. *Transactions on Machine Learning Research*. Cited on pages 32, 86, 116, 118, and 131.

Giraldo, J. H., K. Skianis, T. Bouwmans, and F. D. Malliaros
2023. On the Trade-off between Over-smoothing and Over-squashing in Deep Graph Neural Networks. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, Pp. 566–576. Cited on page 32.

Glorot, X. and Y. Bengio
2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Pp. 249–256. JMLR Workshop and Conference Proceedings. Cited on page 66.

Goodfellow, I., Y. Bengio, and A. Courville
2016. *Deep Learning*. MIT Press. Cited on page 62.

Gori, M., G. Monfardini, and F. Scarselli
2005. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, Pp. 729–734 vol. 2. Cited on page 28.

Grover, A. and J. Leskovec
2016. Node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, Pp. 855–864, New York, NY, USA. Association for Computing Machinery. Cited on page 27.

Guo, Z., Y. Zhang, Z. Teng, and W. Lu
2019. Densely Connected Graph Convolutional Networks for Graph-to-Sequence Learning. *Transactions of the Association for Computational Linguistics*, 7:297–312. Cited on page 116.

Hamilton, W., Z. Ying, and J. Leskovec
2017. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc. Cited on pages 28, 29, 31, 53, 69, 73, 74, and 108.

Hamilton, W. L.
2020. *Graph Representation Learning*, volume 14 of *Synthesis Lectures on Artificial Intelligence and Machine Learning*. Cited on pages 41, 44, 46, and 49.

Hammond, D. K., P. Vandergheynst, and R. Gribonval
2011. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150. Cited on page 68.

He, K., X. Zhang, S. Ren, and J. Sun
2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, Pp. 1026–1034. Cited on page 66.

He, K., X. Zhang, S. Ren, and J. Sun
2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pp. 770–778. Cited on pages 64, 65, 74, and 115.

Herrmann, M., F. J. D. Lange, K. Eggensperger, G. Casalicchio, M. Wever, M. Feurer, D. Rügamer, E. Hüllermeier, A.-L. Boulesteix, and B. Bischl
2024. Position: Why we must rethink empirical research in machine learning. In *Proceedings of the 41st International Conference on Machine Learning*, R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, eds., volume 235 of *Proceedings of Machine Learning Research*, Pp. 18228–18247. PMLR. Cited on pages 33, 35, and 39.

Hochreiter, S.
1998. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 06(02):107–116. Cited on page 60.

Hochreiter, S. and J. Schmidhuber
1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780. Cited on page 63.

Hoogeboom, E., V. G. Satorras, C. Vignac, and M. Welling
2022. Equivariant Diffusion for Molecule Generation in 3D. Cited on pages 29, 137, 138, and 140.

Hornik, K., M. Stinchcombe, and H. White
1989. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366. Cited on pages 58 and 60.

Hu, L., T. Huang, L. Yu, W. Lin, T. Zheng, and D. Wang
2025. Faithful Interpretation for Graph Neural Networks. *Transactions on Machine Learning Research.* Cited on page 65.

Hu, W., B. Liu, J. Gomes, M. Zitnik, P. Liang, V. Pande, and J. Leskovec
2019. Strategies for Pre-training Graph Neural Networks. In *International Conference on Learning Representations.* Cited on page 107.

Huang, G., Z. Liu, L. van der Maaten, and K. Q. Weinberger
2017. Densely Connected Convolutional Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pp. 4700–4708. Cited on page 65.

Huang, L.
2022. *Normalization Techniques in Deep Learning*, Synthesis Lectures on Computer Vision. Cham: Springer International Publishing. Cited on page 66.

Ioffe, S. and C. Szegedy
2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Cited on page 65.

Jain, S. and B. C. Wallace
2019. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, eds., Pp. 3543–3556, Minneapolis, Minnesota. Association for Computational Linguistics. Cited on pages 65 and 79.

Jaiswal, A., P. Wang, T. Chen, J. Rousseau, Y. Ding, and Z. Wang
2022. Old can be Gold: Better Gradient Flow can Make Vanilla-GCNs Great Again. *Advances in Neural Information Processing Systems*, 35:7561–7574. Cited on pages 32, 75, 113, 116, 122, and 131.

Jha, K., S. Saha, and H. Singh
2022. Prediction of protein–protein interaction using graph neural networks. *Scientific Reports*, 12(1):8360. Cited on pages 28 and 83.

Jumper, J., R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis
2021. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589. Cited on pages 28 and 29.

Keriven, N.
2022. Not too little, not too much: A theoretical analysis of graph (over)smoothing. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, eds., volume 35, Pp. 2268–2281. Curran Associates, Inc. Cited on pages 31, 108, and 114.

Kingma, D. P. and J. Ba
2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations*, Y. Bengio and Y. LeCun, eds., Conference Track Proceedings, San Diego, CA, USA. Cited on pages 59 and 66.

Kipf, T. N. and M. Welling
2017. 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. Cited on pages 28, 30, 53, 68, 69, 112, 115, 171, and 174.

Kornblith, S., M. Norouzi, H. Lee, and G. Hinton
2019. Similarity of Neural Network Representations Revisited. In *Proceedings of the 36th International Conference on Machine Learning*, Pp. 3519–3529. PMLR. Cited on pages 123 and 124.

Krizhevsky, A., I. Sutskever, and G. E. Hinton
2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc. Cited on pages 30 and 171.

Kulatilleke, G. K., M. Portmann, R. Ko, and S. S. Chandra
2022. FDGATII: Fast Dynamic Graph Attention with Initial Residual and Identity. In *AI 2022: Advances in Artificial Intelligence*, H. Aziz, D. Corrêa, and T. French, eds., Pp. 73–86, Cham. Springer International Publishing. Cited on page 116.

Lecun, Y., L. Bottou, Y. Bengio, and P. Haffner
1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324. Cited on page 62.

Lee, S. Y., F. Bu, J. Yoo, and K. Shin
2023. Towards Deep Attention in Graph Neural Networks: Problems and Remedies. In *Proceedings of the 40th International Conference on Machine Learning*, Pp. 18774–18795. PMLR. Cited on page 32.

Leskovec, J.
2023. Stanford CS224W: Machine Learning with GRaphs. https://snap.stanford.edu/class/cs224w-2022/. Cited on pages 43 and 50.

Li, G., M. Müller, B. Ghanem, and V. Koltun
2021. Training Graph Neural Networks with 1000 Layers. In *Proceedings of the 38th International Conference on Machine Learning*, Pp. 6437–6449. PMLR. Cited on pages 100, 108, and 174.

Li, L., J. Wang, M. Franklin, Q. Yin, J. Wu, G. Camps-Valls, Z. Zhu, C. Wang, Y. Ge, and M. Reichstein
2023. Improving air quality assessment using physics-inspired deep graph learning. *npj Climate and Atmospheric Science*, 6(1):1–13. Cited on page 29.

Li, Q., Z. Han, and X.-m. Wu
2018. Deeper Insights Into Graph Convolutional Networks for Semi-Supervised Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1). Cited on pages 97, 107, and 114.

Li, Y., R. Zemel, M. Brockschmidt, and D. Tarlow
2016. Gated graph sequence neural networks. In *Proceedings of ICLR'16*. Cited on page 71.

Liu, X., J. Ding, W. Jin, H. Xu, Y. Ma, Z. Liu, and J. Tang
2021. Graph Neural Networks with Adaptive Residual. In *Advances in Neural Information Processing Systems*, volume 34, Pp. 9720–9733. Curran Associates, Inc. Cited on page 116.

Luan, S., C. Hua, Q. Lu, J. Zhu, M. Zhao, S. Zhang, X.-W. Chang, and D. Precup
2022. Revisiting Heterophily For Graph Neural Networks. *Advances in Neural Information Processing Systems*, 35:1362–1375. Cited on page 77.

Luan, S., C. Hua, M. Xu, Q. Lu, J. Zhu, X.-W. Chang, J. Fu, J. Leskovec, and D. Precup
2023. When Do Graph Neural Networks Help with Node Classification? Investigating the Homophily Principle on Node Distinguishability. In *Thirty-Seventh Conference on Neural Information Processing Systems*. Cited on pages 32, 86, 98, and 108.

Luan, S., M. Zhao, X.-W. Chang, and D. Precup
2024. Training Matters: Unlocking Potentials of Deeper Graph Convolutional Neural Networks. In *Complex Networks & Their Applications XII*, H. Cherifi, L. M. Rocha, C. Cherifi, and M. Donduran, eds., Pp. 49–60, Cham. Springer Nature Switzerland. Cited on pages 107 and 114.

Luong, T., H. Pham, and C. D. Manning
2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, L. Màrquez, C. Callison-Burch, and J. Su, eds., Pp. 1412–1421, Lisbon, Portugal. Association for Computational Linguistics. Cited on page 65.

Ma, Y., X. Liu, N. Shah, and J. Tang
2022. Is homophily a necessity for Graph Neural Networks? In *10th International Conference on Learning Representations, ICLR 2022*. Cited on pages 86, 97, and 98.

McPherson, M., L. Smith-Lovin, and J. M. Cook
2001. Birds of a Feather: Homophily in Social Networks. *Annual Review of Sociology*, 27:415–444. Cited on page 50.

Morris, C., M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe
2019. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, volume 33 of *AAAI'19/IAAI'19/EAAI'19*, Pp. 4602–4609, Honolulu, Hawaii, USA. AAAI Press. Cited on page 56.

Murel, J. and E. Kavlakoglu
2023. What Is Regularization? | IBM. https://www.ibm.com/think/topics/regularization. Cited on page 66.

Nguyen, T., M. Raghu, and S. Kornblith
2021. Do Wide and Deep Networks Learn the Same Things? Uncovering How Neural Network Representations Vary with Width and Depth. In *International Conference on Learning Representations.* Cited on page 123.

Oono, K. and T. Suzuki
2019. Graph Neural Networks Exponentially Lose Expressive Power for Node Classification. Cited on pages 31, 75, 84, 107, and 114.

Ou, M., P. Cui, J. Pei, Z. Zhang, and W. Zhu
2016. Asymmetric Transitivity Preserving Graph Embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Pp. 1105–1114, San Francisco California USA. ACM. Cited on page 27.

Page, L., S. Brin, R. Motwani, and T. Winograd
1999. The PageRank Citation Ranking: Bringing Order to the Web. Techreport, Stanford InfoLab. Cited on pages 26 and 27.

Palowitch, J., A. Tsitsulin, B. Mayer, and B. Perozzi
2022. GraphWorld: Fake Graphs Bring Real Insights for GNNs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, Pp. 3691–3701, New York, NY, USA. Association for Computing Machinery. Cited on pages 100, 172, 173, and 174.

Pei, H., B. Wei, K. C.-C. Chang, Y. Lei, and B. Yang
2019. Geom-GCN: Geometric Graph Convolutional Networks. In *International Conference on Learning Representations.* Cited on pages 28, 108, 172, 173, and 174.

Perozzi, B., R. Al-Rfou, and S. Skiena
2014. DeepWalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD

’14, Pp. 701–710, New York, NY, USA. Association for Computing Machinery. Cited on page 27.

Raghu, M., T. Unterthiner, S. Kornblith, C. Zhang, and A. Dosovitskiy
2021. Do Vision Transformers See Like Convolutional Neural Networks? In *Advances in Neural Information Processing Systems*, volume 34, Pp. 12116–12128. Curran Associates, Inc. Cited on page 123.

Rong, Y., W. Huang, T. Xu, and J. Huang
2020. DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. In *International Conference on Learning Representations*. Cited on pages 66 and 116.

Rosenblatt, F.
1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408. Cited on page 58.

Rozemberczki, B., C. Allen, R. Sarkar, and x. Thilo Gross
2021. Multi-Scale attributed node embedding. *Journal of Complex Networks*, 9(1):1–22. Cited on pages 172 and 173.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams
1986. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536. Cited on page 59.

Rusch, T. K., M. M. Bronstein, and S. Mishra
2023a. A Survey on Oversmoothing in Graph Neural Networks. Cited on pages 31, 75, 116, and 122.

Rusch, T. K., B. Chamberlain, J. Rowbottom, S. Mishra, and M. Bronstein
2022. Graph-Coupled Oscillator Networks. In *Proceedings of the 39th International Conference on Machine Learning*, Pp. 18888–18909. PMLR. Cited on pages 116 and 118.

Rusch, T. K., B. P. Chamberlain, M. W. Mahoney, M. M. Bronstein, and S. Mishra
2023b. Gradient Gating for Deep Multi-Rate Learning on Graphs. In *The Eleventh International Conference on Learning Representations*. Cited on pages 32, 113, 116, 117, and 122.

Sanchez-Gonzalez, A., J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. Battaglia
2020. Learning to Simulate Complex Physics with Graph Networks. In *Proceedings of the 37th International Conference on Machine Learning*, Pp. 8459–8468. PMLR. Cited on pages 28 and 83.

Santoro, A., D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap
2017. A simple neural network module for relational reasoning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc. Cited on page 29.

Satorras, V. G., E. Hoogeboom, and M. Welling
2021. E(n) Equivariant Graph Neural Networks. In *Proceedings of the 38th International Conference on Machine Learning*, Pp. 9323–9332. PMLR. Cited on pages 138 and 140.

Saxe, A. M., J. L. McClelland, and S. Ganguli
2014. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *International Conference on Learning Representations (ICLR).* Cited on page 66.

Scarselli, F., M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini
2009. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1):61–80. Cited on pages 28, 30, and 171.

Shchur, O., M. Mumme, A. Bojchevski, and S. Günnemann
2019. Pitfalls of Graph Neural Network Evaluation. Cited on pages 172 and 173.

Shervashidze, N., P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt
2011. Weisfeiler-Lehman Graph Kernels. *Journal of Machine Learning Research*, 12(77):2539–2561. Cited on page 46.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov
2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958. Cited on page 66.

Tang, J., M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei
2015. LINE: Large-scale Information Network Embedding. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, Pp. 1067–1077, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee. Cited on page 27.

Thang, D. C., H. T. Dat, N. T. Tam, J. Jo, N. Q. V. Hung, and K. Aberer
2022. Nature vs. Nurture: Feature vs. Structure for Graph Neural Networks. *Pattern Recognition Letters*, 159:46–53. Cited on page 108.

Thekumparampil, K. K., C. Wang, S. Oh, and L.-J. Li
2018. Attention-based Graph Neural Network for Semi-supervised Learning. Cited on page 69.

Topping, J., F. D. Giovanni, B. P. Chamberlain, X. Dong, and M. M. Bronstein
2022. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations.* Cited on pages 76 and 107.

Ulyanov, D., A. Vedaldi, and V. Lempitsky
2017. Instance Normalization: The Missing Ingredient for Fast Stylization. Cited on page 65.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. ukasz Kaiser, and I. Polosukhin
2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc. Cited on page 65.

Veit, A., M. Wilber, and S. Belongie
2016. Residual networks behave like ensembles of relatively shallow networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, Pp. 550–558, Red Hook, NY, USA. Curran Associates Inc. Cited on page 65.

Veličković, P., G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio
2018. Graph Attention Networks. In *International Conference on Learning Representations.* Cited on pages 28, 31, 53, 65, 69, 108, and 125.

Wang, D., J. Lin, P. Cui, Q. Jia, Z. Wang, Y. Fang, Q. Yu, J. Zhou, S. Yang, and Y. Qi
2019a. A Semi-Supervised Graph Attentive Network for Financial Fraud Detection. In *2019 IEEE International Conference on Data Mining (ICDM)*, Pp. 598–607. Cited on page 29.

Wang, H., R. Yang, and X. Xiao
2024. Effective Edge-wise Representation Learning in Edge-Attributed Bipartite Graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, Pp. 3081–3091, New York, NY, USA. Association for Computing Machinery. Cited on page 29.

Wang, S., B. Z. Li, M. Khabsa, H. Fang, and H. Ma
2020. Linformer: Self-Attention with Linear Complexity. Cited on page 65.

Wang, X., H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu
2019b. Heterogeneous Graph Attention Network. In *The World Wide Web Conference*, WWW '19, Pp. 2022–2032, New York, NY, USA. Association for Computing Machinery. Cited on page 29.

Wang, Y., Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon
2019c. Dynamic Graph CNN for Learning on Point Clouds. *ACM Trans. Graph.*, 38(5):146:1–146:12. Cited on page 71.

Wong, F., E. J. Zheng, J. A. Valeri, N. M. Donghia, M. N. Anahtar, S. Omori, A. Li, A. Cubillos-Ruiz, A. Krishnan, W. Jin, A. L. Manson, J. Friedrichs, R. Helbig, B. Hajian, D. K. Fiejtek, F. F. Wagner, H. H. Soutter, A. M. Earl, J. M. Stokes, L. D. Renner, and J. J. Collins
2024. Discovery of a structural class of antibiotics with explainable deep learning. *Nature*, 626(7997):177–185. Cited on pages 28 and 83.

Wu, F., A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger
2019. Simplifying Graph Convolutional Networks. In *Proceedings of the 36th International Conference on Machine Learning*, Pp. 6861–6871. PMLR. Cited on pages 31, 32, 68, 89, and 112.

Wu, X., A. Ajorlou, Z. Wu, and A. Jadbabaie
2023. Demystifying oversmoothing in attention-based graph neural networks. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Pp. 35084–35106, Red Hook, NY, USA. Curran Associates Inc. Cited on page 32.

Wu, Y. and K. He
2018. Group Normalization. Cited on page 65.

Xu, K., W. Hu, J. Leskovec, and S. Jegelka
2019. How Powerful are Graph Neural Networks? Cited on pages 28, 31, 55, 56, 57, 64, 69, 73, and 108.

Xu, K., C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka
2018. Representation Learning on Graphs with Jumping Knowledge Networks. In *Proceedings of the 35th International Conference on Machine Learning*, Pp. 5453–5462. PMLR. Cited on page 116.

Xu, M., A. Powers, R. Dror, S. Ermon, and J. Leskovec
2023. Geometric Latent Diffusion Models for 3D Molecule Generation. Cited on pages 137 and 142.

Yan, S., Y. Xiong, and D. Lin
2018. Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1). Cited on page 29.

Yan, Y., M. Hashemi, K. Swersky, Y. Yang, and D. Koutra
2022. Two Sides of the Same Coin: Heterophily and Oversmoothing in Graph Convolutional Neural Networks. In *2022 IEEE International Conference on Data Mining (ICDM)*, Pp. 1287–1292. Cited on pages 28, 31, 32, 78, 84, 86, 107, 108, 113, 114, 116, 117, and 131.

Yang, J., J. Lu, S. Lee, D. Batra, and D. Parikh
2018. Graph R-CNN for Scene Graph Generation. In *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, eds., volume 11205, Pp. 690–706, Cham. Springer International Publishing. Cited on page 29.

Yang, Z., W. W. Cohen, and R. Salakhutdinov
2016. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, Pp. 40–48, New York, NY, USA. JMLR.org. Cited on pages 172 and 173.

Ying, C., T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, and T.-Y. Liu
2021. Do Transformers Really Perform Bad for Graph Representation? Cited on pages 29 and 72.

Ying, R., R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec
2018a. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Pp. 974–983, London United Kingdom. ACM. Cited on page 29.

Ying, Z., J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec
2018b. Hierarchical Graph Representation Learning with Differentiable Pooling. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc. Cited on page 53.

You, J., Z. Ying, and J. Leskovec
2020. Design Space for Graph Neural Networks. In *Advances in Neural Information Processing Systems*, volume 33, Pp. 17009–17021. Curran Associates, Inc. Cited on pages 66, 67, 72, 73, 74, 78, 172, and 174.

Yun, S., M. Jeong, R. Kim, J. Kang, and H. J. Kim
2019. Graph Transformer Networks. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc. Cited on page 72.

Zhang, H., Y. Dauphin, and T. Ma
2019. Fixup Initialization: Residual Learning Without Normalization. In *ICLR*. Cited on page 65.

Zhang, K., P. Deidda, D. Higham, and F. Tudisco
2025. Rethinking Oversmoothing in Graph Neural Networks: A Rank-Based Perspective. Cited on page 75.

Zhang, W., Z. Sheng, Z. Yin, Y. Jiang, Y. Xia, J. Gao, Z. Yang, and B. Cui
2022. Model Degradation Hinders Deep Graph Neural Networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, Pp. 2493–2503, New York, NY, USA. Association for Computing Machinery. Cited on page 116.

Zhao, L. and L. Akoglu
2019. PairNorm: Tackling Oversmoothing in GNNs. In *International Conference on Learning Representations*. Cited on pages 28, 108, and 116.

Zhou, D., O. Bousquet, T. Lal, J. Weston, and B. Schölkopf
2003. Learning with Local and Global Consistency. In *Advances in Neural Information Processing Systems*, volume 16. MIT Press. Cited on page 46.

Zhou, J., G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun
2020. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81. Cited on page 74.

Zhou, K., X. Huang, Q. Song, R. Chen, and X. Hu
2022. Auto-GNN: Neural architecture search of graph neural networks. *Frontiers in Big Data*, 5. Cited on pages 74 and 79.

Zhu, J., Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra
2020. Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs. In *Advances in Neural Information Processing Systems*, volume 33, Pp. 7793–7804. Curran Associates, Inc. Cited on pages 86 and 108.

Zhu, X. and Z. Ghahramani
2002. Learning from labeled and unlabeled data with label propagation. Cited on page 46.

Author's Publications

Gomes, D., K. Efthymiadis, A. Nowe, and P. Vrancx
2024a. Depth scaling in graph neural networks: Understanding the flat curve behavior. *Transactions on Machine Learning Research.*

Gomes, D., A. Nowe, and P. Vrancx
2024b. Understanding feature/structure interplay in graph neural networks. In *The Third Learning on Graphs Conference.*

Gomes, D., F. Ruelens, K. Efthymiadis, A. Nowe, and P. Vrancx
2022. When are graph neural networks better than structure-agnostic methods? In *I Can't Believe It's Not Better Workshop: Understanding Deep Learning Through Empirical Falsification @ NeurIPS 2022.*

Appendix

A Methodological Details

A.1 Data of Machine Learning Research Trends

The data used to illustrate the relationship and growth trends between Graph Learning, Deep Learning, and GNNs in Figure 1.2 were obtained using the public tool from Batalha [2025]. This tool enables keyword-based queries over the ArXiv repository to return the number of publications per year for specific research topics.

For each subfield, we defined relevant keywords to capture the relevant body of literature:

- Deep Learning: *"deep learning"*
- Graph Learning: *"graph learning", "graph representation learning", "graph classification", "learning on graphs", "node classification", "link prediction", "edge prediction"*
- GNNs: *"graph neural networks"*

We then retrieved the yearly counts of ArXiv submissions that matched these keywords, compiling time series data that reflect the publishing trends within each research area. The resulting data allowed us to plot the growth of these fields, alongside key foundational works [Scarselli et al., 2009; Krizhevsky et al., 2012; Kipf and Welling, 2017] to contextualize their influence on the pace of research activity. This method and keyword selection ensure that the data captures the key trends for each topic; however, it is important to highlight that it may not account for every existing paper, especially if indirectly related to these fields.

A.2 Experimental Details of Graph Properties vs. GNN Layer Performance Study

This section provides the methodological details for the study summarized in Table 2.2, which investigates the influence of graph structural properties on the performance of different GNN layer types. Specifically, we evaluate four widely used GNN layers—GCN, GAT, GIN, and GraphSAGE—on artificially generated graphs that systematically vary in homophily and edge density.

The artificial graphs are generated using the stochastic block model (SBM) implementation provided by Palowitch et al. [2022]. This framework enables controlled manipulation of selected graph properties while holding others approximately constant; however, it is important to recognize that complete isolation is not fully achievable due to inherent dependencies among properties (e.g., between edge density and clustering coefficient). In our setting, we specifically control homophily and edge density while aiming to minimize unintended variation in other structural characteristics.

Each generated graph comprises 1000 nodes, equally partitioned into two classes, thus defining a balanced binary node classification task. Homophily is varied across three levels: low (< 0.4), medium ($0.4 - 0.6$), and high (> 0.6). Similarly, edge density is categorized into low (order of 10^{-3}), medium (order of 10^{-2}), and high (order of 10^{-1}).

All GNN architectures are implemented with the PyG framework [Fey and Lenssen, 2019], ensuring consistency in implementation. We leveraged GraphGym [You et al., 2020] with its recommended base settings, which include the Adam optimizer with a learning rate of 0.01, ReLU activation functions, no dropout, and a fixed hidden dimension of 8 channels. Each configuration was trained for 100 epochs, with performance reported at the best epoch. For each graph, we performed an 80-10-10% train/validation/test split over multiple independent runs to enable paired significance testing. Each model variant is configured with two GNN layers of the specified type (GCN, GAT, GIN, or GraphSAGE), enabling a fair comparison across methods under the same depth constraint. The performance metric reported is the classification accuracy on the node classification task.

A.3 Benchmarks' Overview

- Citation networks [Yang et al., 2016] are homophilic networks where nodes correspond to scientific papers and links encode citations.
- Coauthor datasets [Shchur et al., 2019] are an homophilic graph problem where edges connect authors (nodes) that coauthored a paper.
- Wikipedia datasets [Rozemberczki et al., 2021] correspond to heteropilic graphs where nodes correspond to web pages and edges to hyperlinks between them.
- WebKB were introduced in [Pei et al., 2019]. Similarly to Wikipedia, these networks also encode web pages and hyperlinks.

Table A.1: Properties of benchmark datasets. Legend: N–Nodes; E–Edges; F–Features; C–Classes; h–Homophily; d_e–Edge Density.

Dataset	**# N**	**# E**	**# F**	**# C**	h	d_e	**Type**
Cora [Yang et al., 2016]	2708	5429	1433	7	.81	.00074	Citation
CiteSeer [Yang et al., 2016]	3327	4732	3703	6	.74	.00043	Citation
Pubmed [Yang et al., 2016]	19717	44338	500	3	.80	.00011	Citation
CS [Shchur et al., 2019]	18333	163788	6805	15	.81	.00049	Coauthor
Physics [Shchur et al., 2019]	34493	495924	8415	5	.93	.00042	Coauthor
Chameleon [Rozemberczki et al., 2021]	2277	36101	2325	5	.23	.0070	Wikipedia
Squirrel [Rozemberczki et al., 2021]	5201	217073	2089	5	.22	.0080	Wikipedia
Actor [Pei et al., 2019]	7600	33544	931	5	.22	.00058	Actor
Cornell [Pei et al., 2019]	183	295	1703	5	.30	.0088	WebKB
Texas [Pei et al., 2019]	183	309	1703	5	.11	.0092	WebKB
Wisconsin [Pei et al., 2019]	251	499	1703	5	.21	.0079	WebKB

- The Actor network was also used in [Pei et al., 2019] and consists of a sub-problem of the film-director-actor-writer network. In this dataset, an actor-only subgraph is considered, where nodes represent actors and edges define co-occurrences in the same Wikipedia page.

A.4 Experimental Details of Feature/Structure Decoupling Study

Artificial Graphs Generation Artificial graphs are generated using the SBM as implemented by Palowitch et al. [2022], which enables control of certain graph properties, namely edge density, homophily, and feature SNR (a metric of feature homogeneity between classes, which equals to 1 in homogeneous scenarios and increases with heterogenity, i.e. as features become more separable). Please refer to Palowitch et al. [2022] for more framework-specific details. We generate graphs with 1000 nodes. Each node is assigned a label to create a class-balanced binary node classification problem, i.e., each graph comprises 500 nodes of each class. All artificially generated graphs are available as supplementary material[1] for repeatability.

Benchmarks Four real-world datasets with different properties are also selected to extend our experiments and for benchmarking purposes. Cora and CiteSeer [Yang et al., 2016] are homophilic citation networks; Chameleon [Rozemberczki et al., 2021] and Texas [Pei et al., 2019] are heterophilic graphs, where nodes correspond to web pages and edges to hyperlinks between them. A complete description of these datasets, including relevant properties, can be found in subsection A.3.

[1]https://github.com/dsg95/decoupling-graph-info

Models and Evaluation We consider the GCN [Kipf and Welling, 2017] as a base for all experiments. As discussed in Section 3, more complex layer and model types are also used to extend our analyses: reversible GCN (RevGCN) [Li et al., 2021], due to its robustness to oversmoothing even for deep GNNs; APPNP [Gasteiger et al., 2018] and FiLM convolution [Brockschmidt, 2020] which seem to perform adequately for input graphs that belong to different parts of the graph properties spectrum proposed in Palowitch et al. [2022], where vanilla-GCNs do not. Reported results refer to performance (accuracy) on the test sets (best epoch on the validation set, averaged over 10 runs). Configuration files are available as supplementary material for repeatability.

Table A.2: Graph convolution layers of RevGCN, APPNP and FiLM. f_θ and g_θ correspond to learnable functions.

Method	Layer
RevGCN [Li et al., 2021]	$\mathbf{H}^l = \hat{\tilde{\mathbf{A}}} \cdot \textit{Dropout(ReLU(Norm(}\mathbf{H}^{l-1}\textit{)))} \cdot \mathbf{W}^l$
APPNP [Gasteiger et al., 2018]	$\mathbf{Z}^l = (1-\alpha)\hat{\tilde{\mathbf{A}}}\mathbf{Z}^{l-1} + \alpha\mathbf{H}, \quad \mathbf{Z}^0 = \mathbf{H} = f_\theta(\mathbf{X}), \quad \alpha \in (0,1]$
FiLM [Brockschmidt, 2020]	$\mathbf{H}^l = \sigma(\gamma^{l-1}\hat{\tilde{\mathbf{A}}}\mathbf{H}^{l-1}\mathbf{W}^l + \beta^{l-1}), \quad \beta^{l-1}, \gamma^{l-1} = g_\theta(\mathbf{H}^{l-1})$

Implementation All neural network architectures consist of a single linear layer for feature transformation into 8 channels, followed by 2 message-passing layers (or fully connected layers for the MLP baseline), and a node classification head (linear layer). We use the PyG library [Fey and Lenssen, 2019] and all experiments are run using GraphGym framework [You et al., 2020], which we extended to include more advanced architectures (new layer types or models) when needed. We performed mini-batch training using neighbourhood sampling (batch size of 32) and considered a train-validation-test split of 80%-10%-10%. All experiments employed GraphGym's recommended base settings: the Adam optimizer with learning rate 0.01, ReLU activations, and no dropout. Models were trained for 100 epochs, and results were recorded at the epoch with the best validation performance. Multiple independent runs were performed to compute mean and standard deviations.

A.5 Experimental Details of GCN+InitRes Depth Scaling Study

Benchmarks We consider the ten dataset partitions of Pei et al. [2019], where each partition consists of randomly splitting the nodes of each class in 60%-20%-20% for training-validation-testing, respectively. An analogous splitting was applied for Coauthor, not considered by Pei et al. [2019]. A complete description of these datasets can be found in subsection A.3.

Training conditions All GNN architectures consist of a single linear layer for input feature transformation into 16 channels, followed by L message-passing layers (or a fixed number of 2 linear layers for the MLP baseline), and a node classification head (linear layer). Besides L and α, all hyperparameters are considered as per Table A.3 shows.

Table A.3: Hyperparameters considered in the GCN+InitRes depth scaling study.

Hyperparameter	**Value**
Network Architecture	
No. Layers (L)	$\{1, 2, 4, 8, 16, 32, 64, 128\}$
Residual strength (α)	$\{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$
Hidden channels	16
Training	
Dropout rate	0.5
Weight decay	0.0005
Max. Epochs	500
Early Stopping (patience)	50
Learning rate	0.01
Optimizer	Adam

This setup means that each $\langle L, \alpha\rangle$ pair defines a different GNN, culminating in a total of 48 architectures. We run this setup for all ten partitions of each dataset as a 10-fold cross-validation process. This results in a total of $48 \times 11 \times 10 = 5280$ experiments, for which we log the following, with respect to the test set: 1) the performance in terms of test accuracy; 2) the hidden embeddings, $\mathbf{H}^l$, of all intermediate node representations after each convolution layer $l \in \{1, ..., L\}$.

Statistical Tests for Performance Comparison We compare the average test accuracy of the different network architectures on each benchmark dataset through statistical tests. Let us consider that each dataset d is associated with a set $A^d_{\langle L,\alpha\rangle} = \{acc_1, ..., acc_{10}\}$, where acc_i refers to the test accuracy of the network defined by $\langle L, \alpha\rangle$ on the i-th partition of d. Let us yet define $\mu^d_{\langle L,\alpha\rangle}$, the average of all elements in $A^d_{\langle L,\alpha\rangle}$. We can finally define $\mu^d_{\mathbf{best}}$ which corresponds to the network architecture leading to the largest average test accuracy in d, $A^d_{\mathbf{best}}$. This set is taken as a reference of the best performance in d, against which all other sets $A^d_{\langle L,\alpha\rangle}$ shall be compared. For that purpose, we conduct a paired t-test on the following hypotheses:

- H_0 (*null hypothesis*): $\mu^d_{\mathbf{best}} = \mu^d_{\langle L,\alpha\rangle}$;
- H_a (*alternate hypothesis*): $\mu^d_{\mathbf{best}} \neq \mu^d_{\langle L,\alpha\rangle}$

We consider a 95% confidence interval, thus rejecting the null hypothesis for p-values < 0.05. This means that for p-values > 0.05 we cannot discard with statistically significant relevance that the averages under comparison are equal, i.e. equivalent to top performance in the cases under analysis.

B Additional Results and Analyses

B.1 Regression Analysis Details

To assess the relationship between the separate contributions of features and structure in GNN performance, we conducted a regression analysis using the metrics C_{struct} and C_{feat}, defined in Section 2 of Chapter 3, as independent variables. The dependent variable was the absolute gain in accuracy of a given GNN model over a feature-only baseline (MLP), which reflects the model's ability to leverage graph-specific information. The complete dataset of input variables (C_{struct} and C_{feat}) and target values (absolute gains) for all tested dataset+GNN combinations is summarized in Table B.4.

Table B.4: Regression: data points (structure and feature components) and target (absolute gains of GNN compared to MLP).

Dataset	Model	C_{struct}	C_{feat}	Absolute Gains
Artificial (informative)	GCN	0.40	0.56	0.07
Artificial (homophily=0.2)	GCN	0.23	0.37	0.04
Artificial (homophily=0.5)	GCN	0.02	0.18	-0.11
Artificial (homophily=0.8)	GCN	0.22	0.42	0.10
Artificial (density=0.003)	GCN	0.16	0.44	0.07
Artificial (density=0.03)	GCN	0.22	0.42	0.10
Artificial (density=0.15)	GCN	0.25	0.35	0.08
Cora	GCN	0.69	0.41	0.14
CiteSeer	GCN	0.64	0.42	0.07
Chameleon	GCN	-0.02	0.34	-0.04
Texas	GCN	0.02	-0.02	-0.16
Artificial (informative)	RevGCN	0.23	0.71	0.04
Cora	RevGCN	0.31	0.75	0.08
CiteSeer	RevGCN	0.25	0.81	0.06
Chameleon	RevGCN	-0.03	0.37	-0.04
Texas	RevGCN	-0.04	0.26	-0.15
Artificial (informative)	APPNP	0.30	0.62	0.03
Cora	APPNP	0.70	0.38	0.15
CiteSeer	APPNP	0.66	0.42	0.08
Chameleon	APPNP	-0.06	0.44	-0.02
Texas	APPNP	0.02	0.06	-0.13
Artificial (informative)	FiLM	0.19	0.73	0.04
Cora	FiLM	0.22	0.72	0.03
CiteSeer	FiLM	0.18	0.78	0.02
Chameleon	FiLM	-0.05	0.41	-0.03
Texas	FiLM	0.02	0.34	0.03

The regression was performed both with and without an interaction term ($C_{struct} \times C_{feat}$) to evaluate whether accounting for joint effects between structure and feature contributions improves the predictive power of the model. The version including the interaction term showed a significantly better fit, as visualized in Figure 3.5 of the main document, and demonstrated stronger statistical relevance, as evidenced by comparing Table B.5 (no interation) and Table B.6 (with interaction).

Table B.5: Regression without interaction: summary of statistics.

Regression Statistics		
	F	*Significance F*
F-Statistic	22.9	3.35e-6
	Coefficients	**P-value**
Intercept	-0.0955	6.93e-4
Structure Component (C_{struct})	0.231	3.21e-5
Feature Component (C_{feat})	0.146	8.14e-3

Table B.6: Regression with interaction: summary of statistics.

Regression Statistics		
	F	*Significance F*
F-Statistic	34.4	1.73e-8
	Coefficients	**P-value**
Intercept	-0.179	6.61e-7
Structure Component (C_{struct})	0.930	7.74e-6
Feature Component (C_{feat})	0.435	7.58e-6
Interaction ($C_{struct} \times C_{feat}$)	-1.80	1.97e-4

B.2 Depth Evaluation: Further Examples

This appendix presents additional results from our depth evaluation protocol. Figures B.1, B.2, and B.3 illustrate the behavior of 16-layer GCN+InitRes, GCNII, and G^2 across all benchmarks corresponding to Cases 1, 2, and 3, respectively. Figure B.4 highlights the severity of pathological behavior in Case 3 benchmarks as network depth increases. Figure B.5 and Figure B.6 demonstrate analogous pathologies in 16-layer GAT+InitRes networks and for a large-scale dataset (*ogbn-arxiv*). These examples can be reproduced using our openly available repository[2], which includes all implementation details.

[2]https://github.com/dsg95/gnn-depth-eval

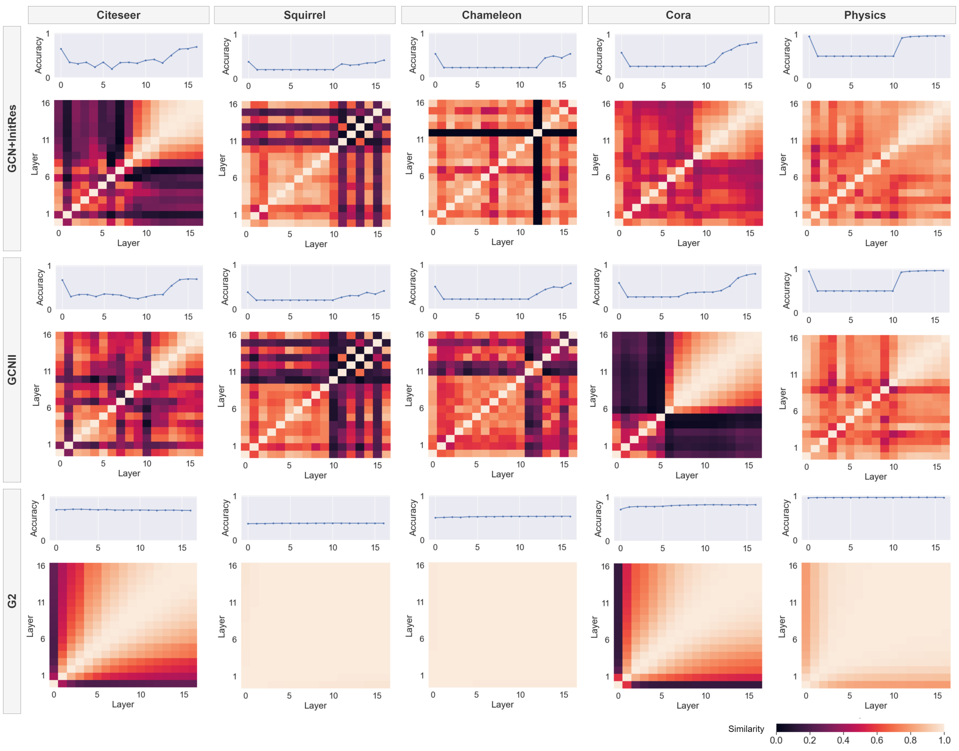


Figure B.1: Results of our depth evaluation protocol for GCN+InitRes, GCNII and G^2 (16-layers) on Case 1 benchmarks (Citeseer, Squirrel, Chameleon, Cora, Physics).

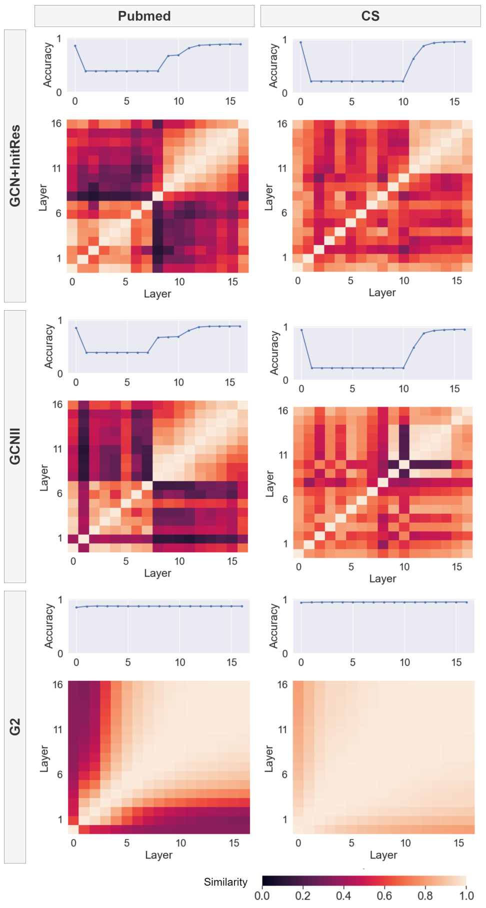


Figure B.2: Results of our depth evaluation protocol for GCN+InitRes, GCNII and G^2 (16-layers) on Case 2 benchmarks (Pubmed, CS).

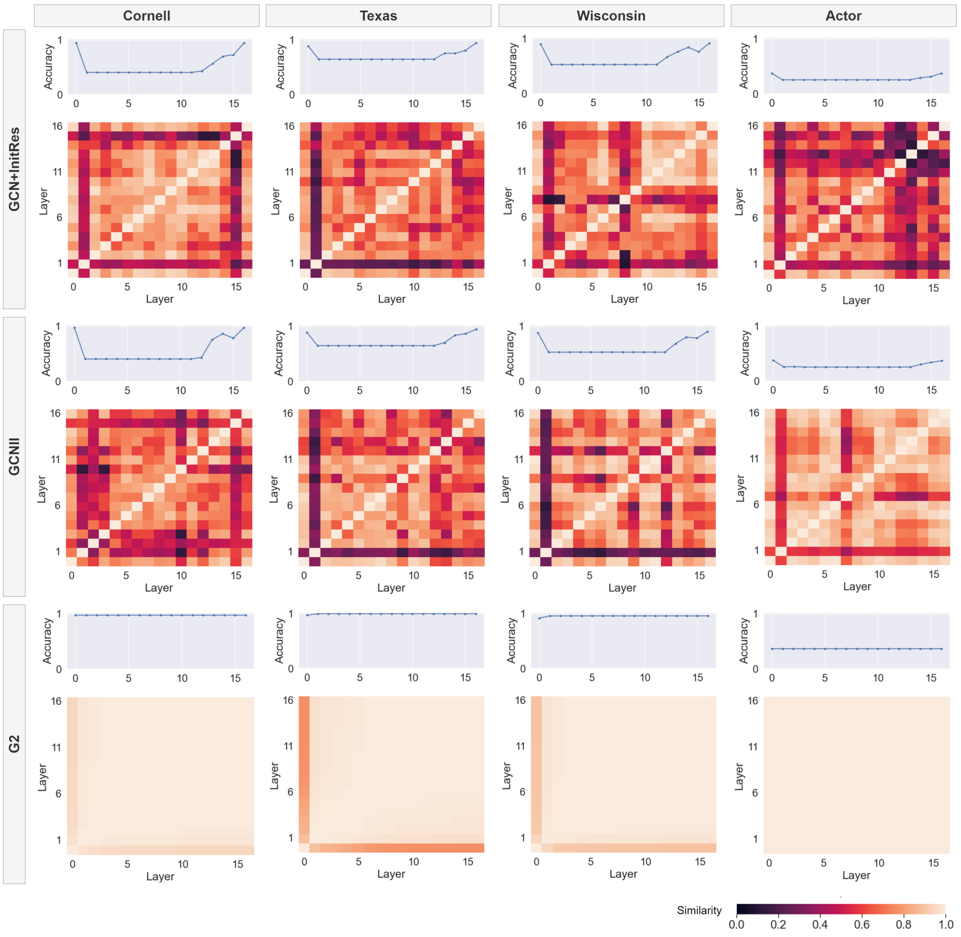


Figure B.3: Results of our depth evaluation protocol for GCN+InitRes, GCNII and G^2 (16-layers) on Case 3 benchmarks (Cornell, Texas, Wisconsin, Actor).

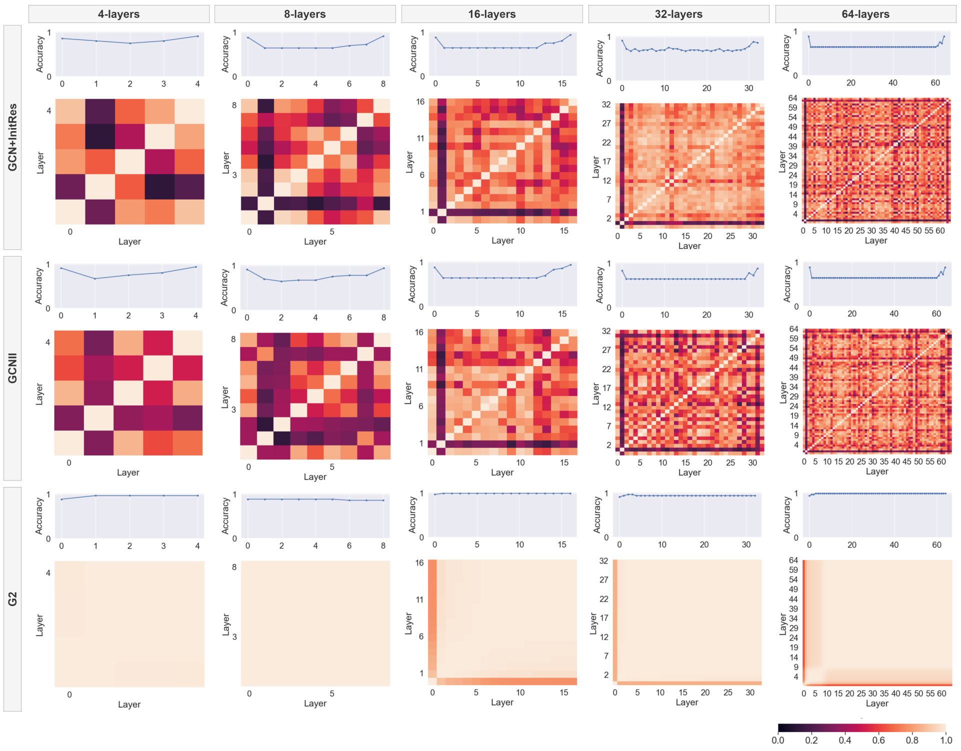


Figure B.4: Results of our depth evaluation protocol for GCN+InitRes, GCNII and G^2 (4/8/16/32/64-layers) on Texas benchmark, as an example of severe cases of Type I and Type II pathologies for all studied depths.

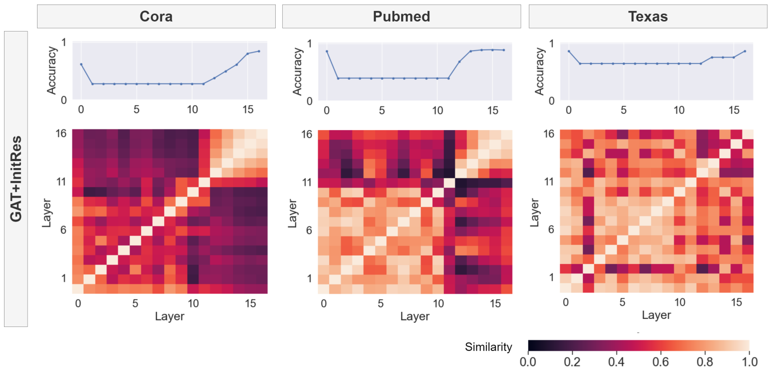


Figure B.5: Results of our depth evaluation protocol for GAT+InitRes (16-layers) on Cora, Pubmed and Texas benchmarks. The implemented architecture of GAT+InitRes is analogous to that of GCN+InitRes (as described in the main manuscript). Similar trends of pathological behavior are verified.

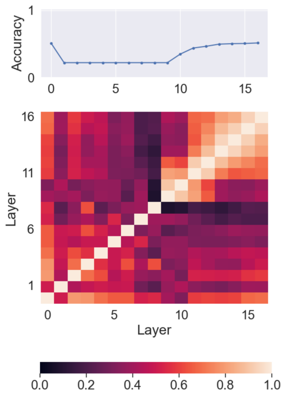


Figure B.6: Results of our depth evaluation protocol for a large scale dataset, *ogbn-arxiv* (16-layers GCNII). Similar trends of pathological behavior are verified for the large scale dataset scenario, comparing to the smaller scale benchmarks.