

UNIVERSITÀ DEGLI STUDI DI SIENA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE E SCIENZE MATEMATICHE



UNIVERSITÀ
DI SIENA
1240

On the Expressive Power of Graph Neural Networks: Beyond the Weisfeiler-Leman Test

Caterina Graziani

PhD in Information Engineering and Science

Supervisors

Prof. Monica Bianchini, Prof. Moreno Falaschi

Examination Committee

Prof. Stefano Melacci

Prof. Barbara Hammer

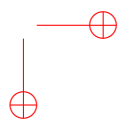
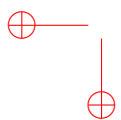
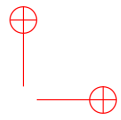
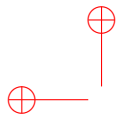
Prof. Fabrizio Costa

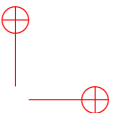
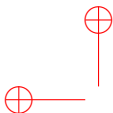
Thesis reviewers

Prof. Filippo Maria Bianchi

Prof. Barbara Hammer

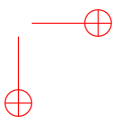
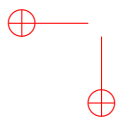
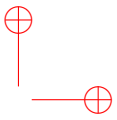
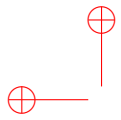
SIENA, 06/07/2024

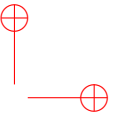
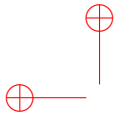




Abstract

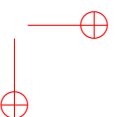
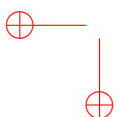
Graphs are pervasive in many applications, hence the quest for models that can learn effective representations of them. Graph Neural Networks (GNNs) have emerged among other architectures for graph processing, and the study of their theoretical properties has recently experienced a rapid rise. A special focus is dedicated to their expressive power, which consists in the ability of GNNs to differentiate non-isomorphic graphs and to approximate functions on graphs. These properties are strictly related to classification/regression tasks where distinguishing between different inputs is crucial for the final performance of the model. In this thesis, we address two main challenges: the lack of expressiveness results for GNNs on arbitrary graph types and the inherently limited expressive power of standard GNNs, which is upper bounded by the Weisfeiler-Leman (1-WL) test. For example, GNNs cannot count any substructure different from star graphs or even distinguish small non-isomorphic graphs. The fields that can benefit from a deeper understanding of the limitations of GNNs are countless, including biology, physics, and social network analysis. In such application domains, the types of graphs involved are varied: for example, data can find natural representations via directed or undirected, homogeneous or heterogeneous, static or dynamic graphs, and even via multigraphs and hypergraphs. However, foundational studies regarding the mentioned graph types are lacking. We aim to fill the gap by providing a comprehensive expressiveness investigation on GNNs for arbitrary graph types. In particular, we devise appropriate extensions of 1-WL and k-WL, identifying relations among them within an algebraic lattice framework. Then, we prove a universal approximation theorem for GNNs across all the aforementioned domains. In particular, the study extends to discrete dynamic graphs, widely used in practice, which require a peculiar analysis approach, due to the difference between dynamic architectures and static ones. We further propose a new, more expressive algorithm, PATH-WL, modifying the aggregation mechanism of GNNs, from the topological neighborhood to a path-based one, enhanced with geodesic distance information. The corresponding model, PAIN (PATH Isomorphism Network), is strictly more expressive than GNNs. Additionally, compared to other architectures known to be more powerful than GNN, we demonstrate that PAIN is not bounded by any of them. We characterize graph classes that can be distinguished by PATH-WL and demonstrate the ability of PATH-WL to count cycles of arbitrary length. This thesis aims to take a step forward in our understanding of GNN capabilities and proposes new strategies to overcome their limitations.





Contents

I	Introduction and Preliminaries	1
1	Introduction	3
1.1	Motivation	5
1.1.1	The Graph Isomorphism Problem	5
1.1.2	The power of the Weisfeiler-Leman test	6
1.1.3	The limits of Graph Neural Networks	7
1.2	Contributions	7
1.3	Outline	9
1.4	List of Publications	9
2	General Preliminaries and Notation	11
2.1	Notation	11
2.2	Graph Theory	11
2.3	Graph Neural Networks	12
2.4	The Expressivity of GNNs	12
2.4.1	The Weisfeiler-Leman test	13
2.4.2	Unfolding tree	14
2.4.3	Higher order versions of the WL test	15
II	Going Beyond Standard Graph-Types	17
3	Weisfeiler and Leman go Edge-Attributed and Dynamic	19
3.1	Preliminary definitions	21
3.2	Edge-Attributed and Dynamic Extensions of 1-WL test and Unfolding Tree	23
3.3	Universal Approximation Capabilities	27
3.3.1	GNNs for Edge-Attributed Static Graphs	27
3.3.2	GNNs for Dynamic Graphs	29
3.4	Experimental Validation	32

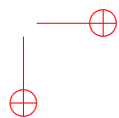
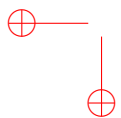
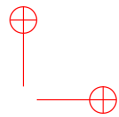
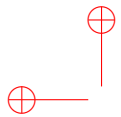


3.5	Conclusion	35
4	The Extended Weisfeiler-Leman Hierarchy for Arbitrary Graph Types	37
4.1	Preliminaries from Lattice Theory	38
4.2	The extended Weisfeiler-Leman Hierarchy	39
4.2.1	Establishing connections between the various tests	40
4.3	Conclusion	42
III	Going Beyond the Topological Neighborhood	43
5	Path-based Graph Neural Networks	45
5.1	Preliminary definitions	46
5.2	Path-WL: A Path-Based WL Test	47
5.3	The Discriminative Power of Paths	50
5.3.1	Relation to the k -WL Hierarchy	50
5.3.2	Counting Cycles	51
5.3.3	One Iteration Is Almost All You Need	53
5.4	Experimental Analysis	55
5.4.1	The PAIN Family	55
5.5	Conclusion	59
IV	Conclusions and Future Perspectives	61
6	General Conclusions	63
	Bibliography	67
A	Other Works	79
	A Neural Network Approach for the Analysis of Reproducible Ribo-Seq Profiles	79
	Point-wise Ribosome Translation Speed Prediction with RNNs	80
	Enhancing Embedding Representations of Biomedical Data using Logic Knowledge	81
	A Two-Stage GAN for Retinal Image Generation and Segmentation	82
	Multi-stage Image Generation for the Semantic Segmentation of Medical Images	82
	Blinking Rate Comparison: Patients with Chronic Pain and Parkinson's Disease	83
B	Appendix	85
B.1	Additional Preliminaries	85
B.2	Universal Approximation Capabilities	86
B.2.1	Proof of Lemma 3.1	86
B.2.2	Proof of Proposition 3.4	87
B.2.3	Proof of Proposition 3.7	87
B.2.4	Proof of Theorem 3.5	88
B.2.5	Proof of Theorem 3.8	89
B.2.6	Proof of Theorem 3.9	92



Contents

B.3	WL Hierarchy	95
B.3.1	Proof of Theorem 4.8	95
C	Appendix	97
C.1	Additional Related work	97
C.2	Additional preliminaries	98
C.3	Path-WL: A path-based WL test	99
C.3.1	Proof of Proposition 5.2	99
C.4	Relation to the k -WL Hierarchy.	100
C.4.1	Proof of Theorem 5.3	100
C.4.2	Proof of Theorem 5.4	100
C.5	Counting cycles	104
C.5.1	Proof of Theorem 5.5	104
C.5.2	Proof of Corollary 5.8	105
C.6	One iteration is almost all you need	105
C.6.1	Proof of Theorem 5.9	105
C.6.2	Proof of Theorem 5.11	108
C.7	Importance of iterations	109
C.8	Additional Information on Experiments	109
	Acknowledgements	113



List of Figures

1.1	(top) Nature’s co-citation network, cover art by A.-L. Barabasi, A. J. Gates, A. Grishchenko, Q. Ke, M. Martino, and O. Varol. 2019. (bottom left) 2011 cover of [1] inspired by network medicine, (bottom right) 2004 cover of [2] focusing on network biology.	4
1.2	The original paper of Weisfeiler and Leman.	6
1.3	Example of non isomorphic 3-regular graphs that 1-WL fails to distinguish.	7
2.1	Two iterations of 1-WL. As a starting point, for $i = 0$, the colors are assigned uniformly to the nodes. From the second iteration ($i = 2$) on, the partition is stable and the algorithm terminates.	13
2.2	The unfolding tree \mathbf{UT}_1^2 for the graph on the left.	14
3.1	a) In Thm. 3.2, we prove the equivalence of the attributed unfolding tree equivalence (AUT) and the attributed 1-WL equivalence (1-AWL) for SAUHG. Afterward in Thm. 3.5, we show a result on the approximation capability of static GNNs for SAUHG (AGNN) using the AUT equivalence. b) Analogously to the attributed case, we show similar results for Dynamic GNNs (DGNN) which can be used on temporal graphs.	20
3.2	DWL test on the graph G composed of three static graph snapshots. Taken from [3].	25
3.3	Graphs G and G' that are AWL equivalent for every snapshot but not DWL equivalent.	26
3.4	These four static graphs are used as components to generate the synthetic dataset. Graphs a) and b) are equivalent under the static 1-WL test; the same holds for c) and d).	33
3.5	Experimental Framework E1. Training accuracy over the epochs for a DGNN trained on the dataset containing dynamic graphs up to time length $T = 4$ (a) and $T = 5$ (b).	34
3.6	Experimental framework E2. Training accuracy a) and training loss b) over the epochs for several DGNNs trained on the dataset containing dynamic graphs up to time length $T = 5$. Figure b) is in logarithmic scale.	34

4.1	The WL complete lattice. Its minimum is the 1-WL test, equivalent to the 2-WL test, and its maximum is the Attributed Graph Isomorphism (AGI) and the Dynamic Graph Isomorphism (DGI) test.	37
4.2	Hasse diagram of the lattices M_3 (left) and N_5 (right).	39
4.3	The graphs a) and b) are falsely recognized as 1-WL and 1-AWL isomorphic; the same holds for the 3-WL and 3-AWL. The graphs b) and c) are falsely recognized as isomorphic just by the 1-WL/3-WL test while the 1-AWL/3-AWL test correctly recognizes both graphs as non isomorphic. The same result holds for the graphs b) and c) for 1-WL and 1-AWL while 3-WL and 3-AWL can distinguish both. The 1-WL and 3-WL equivalences on the graphs from a) and b) and a) and c) without attributes are taken from [4, §3.1].	41
5.1	Stable graph coloring after one iteration of 1-WL. These two not isomorphic graphs represent the two real molecules Decalin and Bicyclopentyl [4]. . .	48
5.2	Example of non-isomorphic graphs for which 1-WL fails, but PATH-WL can distinguish them in the first iteration. In the grey boxes, we visualize the path multisets up to length 5 for nodes v and v'	49
5.3	<i>Rook's</i> 4x4 graph (left) and the <i>Shrikhande</i> (right) graph. They cannot be distinguished by 3-WL but by 0-PATH-WL ^{7,(1)} and 1-PATH-WL ^{4,(1)}	53
5.4	Graphs that cannot be distinguished by 1-WL but can be distinguished by 0-PATH-WL ^ℓ with (a) $\ell = 3$ and (b),(c) $\ell = 4$. The required length ℓ corresponds to the order of the smallest connected component.	54
5.5	Two non-isomorphic 1-WL-equivalent graphs that can be distinguished by 0-PATH-WL ^{2n-1,(1)}	55
5.6	Counterexample that shows that path multisets alone are not more expressive than 1-WL.	55
5.7	Results on SR for path length 4 and marked neighbors. The blue line indicates the failure rate of 3-WL.	58
B.1	The ATTACH operator on trees.	92
C.1	The coloring after one iteration of PATH-WL is enough to distinguish the two non-isomorphic graphs that 1-WL cannot distinguish. Note that the partition is the same but the colors of the nodes are different.	100
C.2	The construction of the homomorphism ϕ from the cycle C_{10} to the 5-clique. The arrows connect the vertices via the function ϕ , which is surjective. The fact that the nodes are connected such that ϕ is a homomorphism, is guaranteed by the presence of the Eulerian cycle in the clique. Indeed, to construct the cycle it is enough to follow the Eulerian cycle on the clique, starting from any edge.	102
C.3	(a) Deleting one vertex from the 6-clique, the red vertex x , results in a 5-clique. (b) A 5-clique is the homomorphic image of a 10-cycle (See C.1 (i), for k odd).	103

List of Figures

C.4 First step of the procedure: double the red node a and open the cycle. The new node is mapped to $\phi(a)$ 103

C.5 Sketch for the proof of Lemma C.2. 105

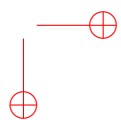
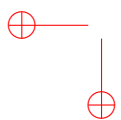
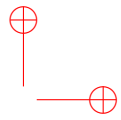
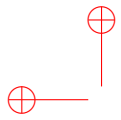
C.6 Example of a degree-invariant transformation. (a) The two graphs G and H can be any two graphs, with at least one edge. (b)–(c) represent the two steps of the transformation: first remove one edge and then link together the two structures in such a way that the degree of the nodes is preserved. 107

C.7 Graphs G and H and the relative multisets of paths of length up to 3, for the node v in G and u in H . These two graphs serve as a counterexample for paths being more discriminative than unfolding trees (and thus 1-WL). 108

C.8 Counterexample showing that **PT**-equivalence is not more expressive than 1-WL. Indeed $v \approx_{WL} u$ but the path-based unfolding trees for u and v are identical. 108

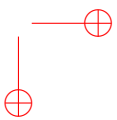
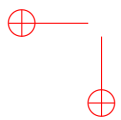
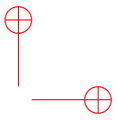
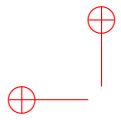
C.9 The two nodes v and u are indistinguishable by 0-PATH-WL^{3,(1)}, indeed they have the same color after iteration 1. But computing the paths with colored nodes allows us to distinguish v and u . That is, 0-PATH-WL _{v} ^{3,(2)} \neq 0-PATH-WL _{u} ^{3,(2)} 109

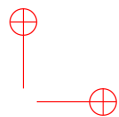
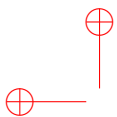
C.10 The two graphs are colored from the first iteration of 0-PATH-WL with paths of length 2. The nodes u and v are not distinguishable, but the multiset of paths contains different paths. The output of the second iteration will result in different colors. That is, 0-PATH-WL _{v} ^{2,(2)} \neq 0-PATH-WL _{u} ^{2,(2)} 110



List of Tables

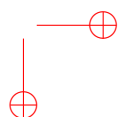
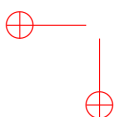
5.1	Summary of our theoretical results.	50
5.2	Mean and standard deviation of accuracy (\uparrow) on the CSL dataset.	57
5.3	Pairs of graphs in EXP that cannot be distinguished by the given models. Bold marks the strongest result.	57
5.4	Mean and standard deviation of mean absolute error on the ZINC (12,000 nodes with edge features) dataset. GIN, GCN and GAT experiments where conducted by [5], all other GNNs where benchmarked by the cited authors. Bold marks the strongest architecture.	58
C.1	Average training time for different GNNs on ZINC. All models were trained with early stopping.	111

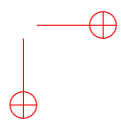
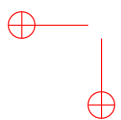
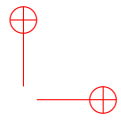
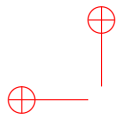


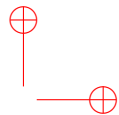
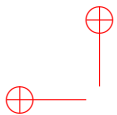


Part I

Introduction and Preliminaries







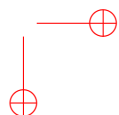
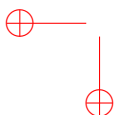
Chapter 1

Introduction

In the past few years, deep learning has led to significant breakthroughs in fields such as computer vision, employing Convolutional Neural Networks (CNNs), or natural language processing, with Recurrent Neural Networks (RNNs) [6, 7] and, recently, with Transformers [8, 9]. However, these techniques are tailored for images and sequential data, and do not perform as well with other data types. As a result, there has been a growing interest in exploring the so-called geometric deep learning [10], namely deep learning for non-Euclidean data such as manifolds or graphs.

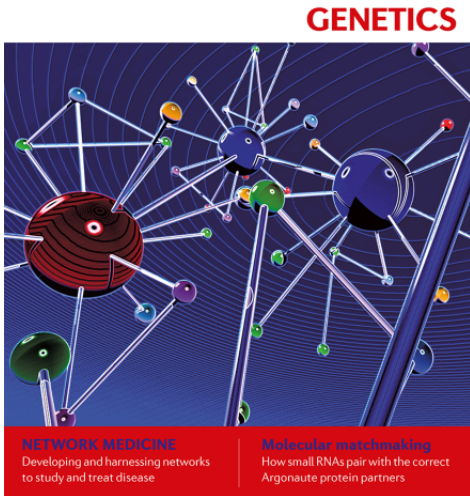
Graphs are complex data structures that relate entities (the nodes) through relations (the edges). They are ubiquitous as they can be used to model several application domains such as social networks [11], traffic networks [12], sensors networks [13], metabolic and regulatory networks [14]. We can represent meshed surfaces as graphs, or complex structures like molecules and compounds, among others. Graphs can also be considered as a generalization of rather regular structures like texts (line graphs), or images (grid-like graphs). Notably, the leading journal *Nature* has devoted exceptional attention to the impact of graphs, dedicating several covers to the topic (see Figure 1.1). Due to the versatility of graphs to model several real-world scenarios, they play a key role in modern machine/deep learning. Still, they are hard to process because of their non-Euclidean structure, which means that graphs generally lack a grid-like form, a common system of coordinates, or a vector space structure. Furthermore, the classical assumption of independence among the entities is negated, given their strong interconnection. Hence, the primary challenge in this domain is to find an effective representation of the graph so that it can be readily utilized by machine learning models [15]. Since 2008, there has been an increasing interest in Graph Neural Network (GNN) approaches for directly processing graphs [16, 17, 18] and many GNN variants have been developed, including Neural Networks for graphs [19], Gated Sequence Graph Neural Networks [20], Spectral Networks [21], Graph Convolutional Neural Networks [22], GraphSAGE [23], Graph Attention Networks [24], and Graph Networks [25].

Despite GNNs have revolutionized learning on graphs and achieved state-of-the-art performance in numerous tasks, theoretical studies have shown that they can fail to solve simple problems [26]. Actually, most studies on the expressive power of GNNs focus on their ability to distinguish non-isomorphic graphs (*separation power*) and approximate functions (*approximation capability*). In fact, the ability of GNNs to distinguish graphs is also related to their capability to approximate functions on graphs. Broadly speaking, a GNN takes a graph as input and produces an output at the node level or for the whole graph, based on the local graph topology. A GNN successfully distinguishes between two





nature
REVIEWS
January 2011 volume 17 no. 1
www.nature.com/reviews



nature
REVIEWS
February 2004 volume 5 no. 2
www.nature.com/reviews

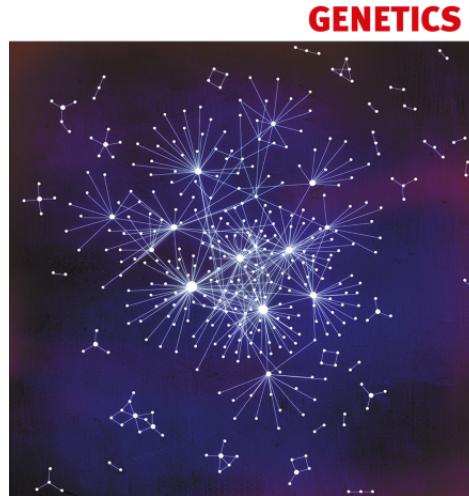


Figure 1.1: (top) Nature's co-citation network, cover art by A.-L. Barabasi, A. J. Gates, A. Grishchenko, Q. Ke, M. Martino, and O. Varol. 2019. (bottom left) 2011 cover of [1] inspired by network medicine, (bottom right) 2004 cover of [2] focusing on network biology.

graphs, if it produces different outputs for each of them. A notable issue occurs when a GNN produces identical outputs for two non-isomorphic, and thus distinct, graphs. Classical approximation results for neural networks such as [27, 28] do not directly extend to GNNs [29, 30, 31] because of the additional requirement of *permutation invariance*. Such requirement ensures that the graph-level outcome is invariant to the order of the

nodes in the graph [25]. This property is crucial for maintaining the consistency of graph-based learning, but it also imposes significant constraints on the types of functions that GNNs can approximate. Indeed, [32] proved that GNNs can approximate any function that is invariant or equivariant to graph isomorphism.

In this chapter, we first discuss the limitations of GNNs, serving as motivation driving the presented research. Then, we detail the contributions made to the field and outline the structure and organization of the thesis. Finally, we present the complete list of publications from which the thesis is drawn, along with additional works that, while not closely related to the topic discussed here, have been developed during my PhD.

1.1 Motivation

The main purpose of this thesis is the analysis of some theoretical properties of graph Neural Networks, which have recently been the subject of an in-depth study [33]. In particular, we focus on their expressive power, which is the ability of GNNs to distinguish graphs and approximate functions on them. But how easy is it to distinguish two graphs, in general?

1.1.1 The Graph Isomorphism Problem

The study of expressive power of GNNs is related to a number of long-standing difficult tasks in graph theory [34, 35], one of the most prominent being the Graph Isomorphism (GI for short). It was recognized as a "disease" by [36] in 1977, and it consists of deciding whether two graphs are isomorphic or not. Two graphs are isomorphic if there is a bijection ϕ between the two node sets that preserves the "adjacencies", that is, if an edge (u, v) exists in the first graph, there exists a link between the corresponding nodes $\phi(u), \phi(v)$ in the second graph. Two isomorphic graphs are essentially the same graph, modulo a suitable permutation of the nodes. Therefore, given the bijection ϕ , it is possible to verify in polynomial time whether two graphs are isomorphic or not, i.e. the GI problem falls into the NP complexity class, thus being of particular theoretical interest. Its actual complexity is still unknown: since 1979, when Garey and Johnson included GI in a list of natural problems whose complexity was unknown [37], it is not known whether a polynomial-time algorithm that solves GI exists nor if it is NP-hard¹. More precisely, GI is believed to fall into the NP-intermediate class, i.e. the class of problems that are in NP but are neither NP-complete nor belong to P.

This problem appears to be highly relevant also in a broad range of applications, from the classification of molecular graphs in chemistry to computational graphs — that can be used to represent mathematical expressions — in computer science. Due to its importance, many efforts have been dedicated to design efficient algorithm to approximate GI [38]. The history of approaches for solving the problem approximately dates back to the 1960s [39, 40, 41, 42]. In 2015, Babai showed an effective algorithm to solve GI in quasi-polynomial $(2^{\log(n)^{O(1)}})$ time [43], improving the previous best bound (from 1983) of

¹Note that both might be true at the same time, since P could coincide with NP.

$\exp^{O(\sqrt{n \log(n)})}$, where n is the number of nodes. The existence of a quasi-polynomial time algorithm for solving GI is one of the strongest indicators against its NP-completeness. In fact, the NP-completeness of GI would imply that every problem in NP is solvable in quasi-polynomial time [44]. Various heuristics to test isomorphism between two graphs are surveyed in [45]. Among them, the Weisfeiler-Leman² (1-WL) method was considered as one possible candidate to solve the GI problem in polynomial time. Actually, isomorphism for many classes of graphs (e.g. trees, planar graphs [46, 47], or graphs of bounded maximum degree) can be decided in polynomial time and, in practice, the graph isomorphism problem can often be solved efficiently [48, 49].

1.1.2 The power of the Weisfeiler-Leman test

The Weisfeiler-Leman test first appeared in 1968 in a bulletin of information sciences of the Soviet Union [50] (see Figure 1.2). Similar methods had already been used in Chemoinformatics since 1965, to create descriptors or *fingerprints* for small molecules [51, 52]. These were employed to determine structural similarities between molecules or to explain their chemical properties [53]. The original paper described the 2-WL variant, which is however equivalent to 1-WL, also known as *Colour Refinement* and it consists of a polynomial-time heuristic for testing graph isomorphism [39, 40]. It iteratively produces a node coloring based on the current color of the node and the colors of its neighbors. After termination, it compares two graphs based on the resulting color sets of their nodes.

If they are different, the two graphs are not isomorphic, otherwise they are *possibly* isomorphic. 1-WL provides only a necessary condition to isomorphism.

In [48], it was shown that 1-WL identifies *almost all* graphs after two iterations. More specifically, the fraction of graphs of order n which are not identified by 1-WL tends to 0 as n tends to infinity [54]. This algorithm is also applied to speed up computations in other fields, for example in static program analysis [55] or within the context of graph kernels in machine learning [56]. A comprehensive description of power and limits of 1-WL can be found in [57, 44, 58].

We are particularly interested in the relation between 1-WL and GNNs. Independently, [26] and [59] showed that the separation power of GNNs is characterized by 1-WL. Equivalently, the 1-WL test exactly defines the classes of

Figure 1.2: The original paper of Weisfeiler and Leman.

graphs that GNNs can recognize as non-isomorphic. A deep understanding of the relationship between 1-WL and GNNs is fundamental to study the theoretical properties

²We use the spelling “Leman” here as Andrew Leman, co-inventor of the algorithm, preferred it over the transcription “Lehman”; see <https://www.iti.zcu.cz/wl2018/pdf/leman.pdf>.

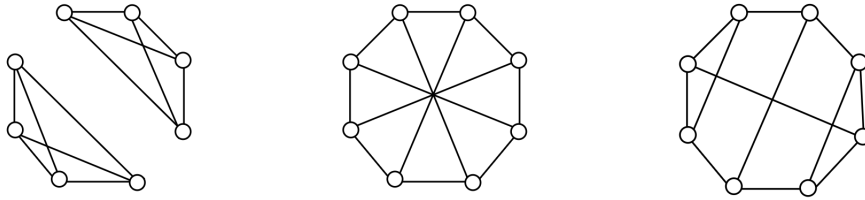


Figure 1.3: Example of non isomorphic 3-regular graphs that 1-WL fails to distinguish.

of the latter and goes beyond expressiveness. For example, [60] derived a generalization bound for GNNs, relating their VC dimension with the number of colors produced by the 1-WL.

1.1.3 The limits of Graph Neural Networks

Graph Neural Networks are a large class of relational models for graph processing. Once proper weights are learned, GNNs are at most as expressive as 1-WL. This correspondence justifies the effectiveness of GNNs in many tasks but also clarifies its limitations. For example, 1-WL systematically fails to distinguish simple graphs (see Figure 1.3). In [61], several non-isomorphic graph pairs are described that are not distinguishable by 1-WL test and hence by GNNs. This limitation can be partially attributed to the fact that 1-WL can only count simple substructures like star graphs [62, 63]. In fact, 1-WL cannot recognize or count cycles [57] and this has important consequences for GNNs in terms of applications. For example, in Bioinformatics, the properties of a chemical molecule may depend on the presence or absence of small substructures [64]; in social network analysis, the count of triangles allows to evaluate the presence and the maturity of communities [65, 66, 67, 68, 69]. Another structural property that 1-WL fails to detect is the presence of disconnected components, which is crucial in many applications, ranging from social networks to biochemistry [70, 71].

Concerning approximation capabilities, GNNs turned out to be universal approximators on graphs *modulo* the constraints enforced by 1-WL. A limit of current results is that they only apply to static undirected homogeneous graphs with node attributes. In contrast, real-life applications often involve a much larger variety of graph types, and the usual results on expressivity cannot be trivially extended to distinguish them.

1.2 Contributions

This thesis focuses on knowledge gaps in the theoretical understanding of the expressive power of GNNs. The goal is to fill these gaps and use the knowledge gained to design new GNN models. In particular, we study the GNN expressive capability on graph domains that have not yet been considered, also providing new tools to better understand the GNN expressiveness in general. Moreover, we propose a novel GNN architecture using paths to extend their expressive power and study such architecture from a theoretical and

experimental viewpoint.

The main contributions of the thesis are listed in the following.

- We extend the expressiveness results on GNNs to *all* graph types mentioned in the previous section. To this aim, we consider a large class of graphs, called Static Attributed Undirected Homogenous Graphs (SAUHGs), and we extend the Weisfeiler-Leman test definition to both SAUHGs and dynamic graphs. Moreover, we consider an equivalent way to test isomorphism, other than 1-WL, that is the unfolding tree. An unfolding tree consists of a tree constructed by a breadth-first visit of the graph, starting from a given node. If the unfolding trees of two nodes are equal in the limit, the nodes are called unfolding tree equivalent. Two graphs are possibly isomorphic if all their unfolding trees are equal. From [72], it is known that for static undirected node-attributed graphs, both the unfolding tree and the Weisfeiler–Leman approach for testing the isomorphism of two graphs are equivalent. We show that this equivalence holds also for the devised extensions. Another main result consists in the analysis of the approximation capabilities of generic GNNs for dynamic graphs and SAUHGs. We show that they are capable of approximating, in probability and up to any precision, any measurable function on graphs that respects the corresponding 1-WL/unfolding equivalence. Moreover, the proof is constructive and allows us to deduce hints on the GNN architecture that can achieve the desired approximation. Finally, we provide an experimental validation of our theoretical results on synthetic datasets. See [3] for more details.
- We extend the higher-order WL test (k -WL) to arbitrary graph types, and we study the relations among the various tests. The result is a complete, distributive, and modular lattice. Interestingly, this draws a connection with the field of *General Algebra*, which we aim to explore in more detail in the future. This preliminary work has been presented at the ECML 2022 MLG Workshop [73].
- To overcome the limits of GNNs, we propose PATH-WL, a general class of color refinement algorithms based on paths and geodesic distance information. We study the relation of PATH-WL with 1-WL and also compare PATH-WL to other more expressive architectures present in the literature. We conclude that PATH-WL is strictly more expressive than 1-WL and it is not comparable to other higher-order extensions. A key feature of the effectiveness of PATH-WL is the ability to count cycles of arbitrary length. Moreover, we propose PAIN, a GNN with an expressive power equivalent to PATH-WL, and empirically verify our theoretical results, in both synthetic and real-world datasets. This work has been accepted at ICML 2024 [74] and already presented, in short form, at the Neurips 2023 GLFrontiers Workshop [75].

1.3 Outline

The thesis is organized as follows. Chapter 2 provides all the necessary preliminaries; moreover, to make reading easier, all the notation symbols are collected in a table. Instead, chapter-specific preliminaries will be found at the beginning of each chapter. The second part of the thesis (Part II) is characterized by the extensions to arbitrary graphs of the above described literature results. In particular, Chapter 3 contains the extension of WL and unfolding tree to edge-attributed and dynamic graphs and the approximation theorems for the related GNNs. Chapter 4 proposes an extended hierarchy of WL for arbitrary graphs, related to higher-order models. The third part of the thesis (Part III) consists of the definition of a new framework where the locality of the classical message-passing algorithm is overcome, in favor of a more expressive class of models based on paths (Chapter 5). Conclusions are collected in Part IV while other works, which have characterized my PhD journey, are summarized in Appendix A. Appendices B and C gather proofs, additional details, and insights related to Part II and Part III, respectively.

1.4 List of Publications

My doctoral program has led to the following publications. The symbol * stands for 'equal contribution'.

- **Weisfeiler–Lehman goes Dynamic: An Analysis of the Expressive Power of Graph Neural Networks for Attributed and Dynamic Graphs**
Silvia Beddar-Wiesing*, Giuseppe Alessio D’Inverno*, **Caterina Graziani***, Veronica Lachi*, Alice Moallem-Oureh*, Franco Scarselli, Josephine M. Thomas
Neural Networks, Volume 173, 2024, 106213, ISSN 0893-6080 ([3])
- **On the Extension of the Weisfeiler-Lehman Hierarchy by WL Tests for Arbitrary Graphs**
Silvia Beddar-Wiesing*, Giuseppe Alessio D’Inverno*, **Caterina Graziani***, Veronica Lachi*, Alice Moallem-Oureh*, Franco Scarselli
18th International Workshop on Mining and Learning with Graphs - ECMLPKDD 2022 ([73])
- **The Expressive Power of Path based Graph Neural Networks**
Caterina Graziani*, Tamara Drucks*, Fabian Jogl, Monica Bianchini, Franco Scarselli, Thomas Gärtner
Accepted at ICML2024 ([74]); *Accepted at NeurIPS 2023 GLFrontiers Workshop* ([75])

During the formation period of my PhD, I have produced other works that are not included in the core of the thesis. They are listed below and described in Appendix A in more detail.

- **Enhancing Embedding Representations of Biomedical Data using Logic Knowledge**

Michelangelo Diligenti, Francesco Giannini, **Caterina Graziani**, Stefano Fioravanti, Moreno Falaschi, Giuppe Marra

In 2023 International Joint Conference on Neural Networks (IJCNN) (pp.1-8).

New York, USA : IEEE [10.1109/IJCNN54540.2023.10191706] ([76])

- **Multi-stage Synthetic Image Generation for the Semantic Segmentation of Medical Images**

Paolo Andreini, Simone Bonechi, Giorgio Ciano, **Caterina Graziani**, Veronica Lachi, Natalie Nikolouloupoulou, Monica Bianchini, Franco Scarselli

In Artificial Intelligence and Machine Learning for Healthcare (pp. 79-104).

Cham, Switzerland: Springer [10.1007/978-3-031-11154-9_5] ([77])

- **A Neural Network Approach for the Analysis of Reproducible Ribo-Seq Profiles**

Giorgia Giacomini*, **Caterina Graziani***, Veronica Lachi*, Pietro Bongini*, Niccolò Pancino*, Monica Bianchini, Davide Chiarugi, Angelo Valleriani, Paolo Andreini*

Algorithms, 15(8) [10.3390/a15080274].

Basel, Switzerland: Molecular Diversity Preservation Int. ([78])

- **Blinking Rate Comparison Between Patients with Chronic Pain and Parkinson's Disease**

Emanuel Stefanescu, Niccolò Pancino, **Caterina Graziani**, Veronica Lachi, Maria Lucia Sampoli, Giovanna Maria Dimitri, Alessia Bargagli, Dario Zanca, Monica Bianchini, Dafin Mureşanu, Alessandra Rufa

European Journal of Neurology 29, 669-669, 2022.

Oxford, GB: Blackwell Science. ([79])

- **A mixed statistical and machine learning approach for the analysis of multimodal trail making test data**

Niccolò Pancino*, **Caterina Graziani***, Veronica Lachi*, Maria Lucia Sampoli, Emanuel Ştefănescu, Monica Bianchini, Giovanna Maria Dimitri*

Mathematics 9, 3159, 2021.

Basel, Switzerland: MDPI AG, 2013 [10.3390/math9243159]. ([80])

- **A Two-Stage GAN for High-Resolution Retinal Image Generation and Segmentation**

Paolo Andreini, Giorgio Ciano, Simone Bonechi, **Caterina Graziani**, Veronica Lachi, Alessandro Mecocci, Andrea Sodi, Franco Scarselli, Monica Bianchini

Electronics 11, 60, 2021.

Basel, Switzerland: MDPI [10.3390/electronics11010060]. ([81])

- **Point-Wise Ribosome Translation Speed Prediction with Recurrent Neural Networks**

Pietro Bongini, Niccolò Pancino, Veronica Lachi, **Caterina Graziani**, Giorgia Giacomini, Paolo Andreini, Monica Bianchini

Mathematics, MDPI, 12, 3, 2024. [10.3390/math12030465] ([82])

Chapter 2

General Preliminaries and Notation

This chapter collects the general preliminaries, namely the basic concepts that are common to the whole thesis.

2.1 Notation

We first provide a table gathering the mathematical notation used across the thesis.

\mathbb{N}	natural numbers
\mathbb{R}	real numbers
\mathbb{Z}	integer numbers
\leq	less than or equal (classical order on numbers)
$[n], n \in \mathbb{N}$	sequence $1, 2, \dots, n$
$\mathbb{N}^k, \mathbb{R}^k, \mathbb{Z}^k$	k -dimensional vector spaces
$\mathbf{x} = (x_i)_{i \in [n]}$	ordered sequence (or vector) of n elements
\mathbf{A}	matrix
$\ \cdot\ $	norm on \mathbb{R}
$\ \cdot\ _\infty$	∞ -norm on \mathbb{R}
\emptyset	empty set
\perp	undefined; non-existent element
$\{\cdot\}$	set
$\{\!\!\{\cdot\}\!\!\}$	multiset
$ M $	cardinality of a set M
\cup	union of two (multi)sets
\times	cartesian product of two (multi)sets
\subseteq	subset (included or equal)
\sqsubseteq	less expressive than or equally expressive to
\sim_r	equivalent w.r.t. relation r

2.2 Graph Theory

Let $G = (\mathcal{V}, \mathcal{E})$ be a **graph** with node set \mathcal{V} and edges \mathcal{E} . We refer to the number of nodes $|\mathcal{V}|$ in G as the *order* of G . Let $\mathcal{N}(v)$ be the *neighborhood* of a node $v \in \mathcal{V}$, i.e. the set of all nodes adjacent to v , and be $\delta(v)$ the *degree* of a node $v \in \mathcal{V}$, i.e., the number of neighbors $|\mathcal{N}(v)|$. For a graph G , we denote with Δ the maximum vertex degree of the

graph. In the present work, we only consider finite, homogeneous, and undirected graphs.

In this framework, we consider graphs endowed with a node labeling, which is defined in the following.

Definition 2.1 (Node labeling). Let $G = (\mathcal{V}, \mathcal{E})$ be a graph. We define a **node labeling** as a function $\alpha : \mathcal{V} \rightarrow \Sigma$ with an arbitrary codomain Σ that we call *set of labels*.

We denote a graph endowed with a node labeling by $G = (\mathcal{V}, \mathcal{E}, \alpha)$ and $\alpha(v)$ by α_v for $v \in \mathcal{V}$.

We sometimes refer to α as the node attributes (see Chapter 3 for example) or, in the context of neural networks, as the node features or even as colors, when it comes to coloring procedures.

2.3 Graph Neural Networks

Message-passing graph neural networks (GNNs) leverage the graph structure and the node labeling α to learn a representation vector (or *embedding*) for individual nodes, denoted by \mathbf{h}_v , or for the entire graph, denoted by \mathbf{h}_G . Each node embedding is updated by aggregating the embeddings of its neighboring nodes.

Definition 2.2 (GNN). Let $G = (\mathcal{V}, \mathcal{E}, \alpha)$ be a graph with node labeling α . We initialize the feature of node v as $\mathbf{h}_v^{(0)} = \alpha_v$ for all $v \in \mathcal{V}$. The computation scheme of a message-passing graph neural network for iteration $i > 0$ is defined as

$$\mathbf{h}_v^{(i)} = \text{COMBINE} \left(\mathbf{h}_v^{(i-1)}, \text{AGGREGATE} \left(\left\{ \left\{ \mathbf{h}_u^{(i-1)} : u \in \mathcal{N}(v) \right\} \right\} \right) \right),$$

where $\mathbf{h}_v^{(i)}$ is the feature vector of node v at the i -th iteration. The GNN output for a node-level learning task after iteration k is given by

$$\mathbf{h}_v = \text{READOUT} \left(\mathbf{h}_v^{(k)} \right),$$

and the output for a graph-level learning task is given by

$$\mathbf{h}_G = \text{READOUT} \left(\left\{ \left\{ \mathbf{h}_v^{(k)} : v \in \mathcal{V} \right\} \right\} \right).$$

In [26], it is shown that with a sufficient number of GNN layers and injective COMBINE, AGGREGATE, and READOUT functions the resulting GNN architecture is as expressive as the Weisfeiler-Leman test. Moreover, the depth of the GNN equals the number of iterations of the test. For more details both on foundations and applications of GNNs, we point to recent surveys [83, 84] and to the review of GNN methods [85].

2.4 The Expressivity of GNNs

In this section of the preliminaries, we introduce the tools to study the expressive power of GNNs, namely the Weisfeiler-Leman test, its k -dimensional extension, and the concept of the unfolding tree of a node.

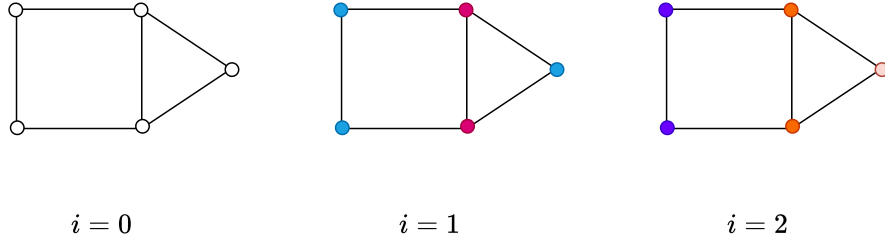


Figure 2.1: Two iterations of 1-WL. As a starting point, for $i = 0$, the colors are assigned uniformly to the nodes. From the second iteration ($i = 2$) on, the partition is stable and the algorithm terminates.

2.4.1 The Weisfeiler-Leman test

We first formally define the Weisfeiler-Leman test, a coloring procedure employed to test graph isomorphism [39, 40].

Definition 2.3 (1-WL). Let $G = (\mathcal{V}, \mathcal{E}, \alpha)$ be a graph with node labeling α and let Σ be a set of colors. Let $\mathbf{c}_v^{(i)} \in \Sigma$ be the color of the node v at iteration i and let $\mathbf{c}_v^{(0)} = \alpha_v$. The Weisfeiler-Leman test updates the color of node v at each iteration $i > 0$ as follows:

$$\mathbf{c}_v^{(i)} = \text{HASH}\left(\mathbf{c}_v^{(i-1)}, \left\{ \left\{ \mathbf{c}_u^{(i-1)} : u \in \mathcal{N}(v) \right\} \right\}\right),$$

where HASH bijectively maps its input to a color from Σ .

1-WL partitions the nodes of a graph into equivalence classes (i.e., the colors), where equivalent nodes are assigned the same color¹. The algorithm terminates with a *stable* coloring when the partitioning does not change between iterations. To test whether two graphs are isomorphic, 1-WL is applied to both graphs. If the stable coloring of the two graphs differs, i.e., the graphs have a different number of nodes with the same color, the graphs are non-isomorphic. The algorithm is not conclusive if the stable coloring is the same, i.e., the two graphs *can be*, but are not guaranteed to be, isomorphic. See Figure 2.1 for an example of 1-WL.

Remark 2.1. Compare Definition 2.2 and Definition 2.3. GNNs can be viewed as a *neural version* of the 1-WL algorithm, where colors are replaced by continuous feature vectors and neural networks are used to aggregate over node neighborhoods [23, 86, 59].

Remark 2.2 (Notation). In general, given a coloring algorithm T , nodes u and v are called *T -equivalent* if they result in the same color after the termination of the algorithm T . It is denoted by $u \sim_T v$. Similarly, graphs G and H are *T -equivalent* if every node v in G is bijectively mapped to a node u in H s.t. $u \sim_T v$. We denote with \mathcal{P}_T the partition in equivalence classes induced by T . Let T_1 and T_2 be two coloring algorithms. We write $T_1 \sqsubseteq T_2$ if \mathcal{P}_{T_2} is finer than or equal to \mathcal{P}_{T_1} and every class in \mathcal{P}_{T_1} is a union

¹Indeed, it is an *equivalence* relation, namely it is reflexive, symmetric, and transitive.

of classes in \mathcal{P}_{T_2} . If we identify the expressivity of an algorithm with its ability to assign different colors to non-isomorphic graphs, the finer the partition, the greater the ability to distinguish. Then, we interpret $T_1 \sqsubseteq T_2$ as T_2 is not less expressive than T_1 . We write $T_1 \sqsubset T_2$ if T_2 is strictly more expressive than T_1 and $T_1 \equiv T_2$ iff $T_1 \sqsubseteq T_2$ and $T_2 \sqsubseteq T_1$. If it holds that $T_1 \not\sqsubseteq T_2$ but at the same time $T_2 \not\sqsubseteq T_1$, the two tests are said to be *incomparable*.

2.4.2 Unfolding tree

Another equivalent way of testing the isomorphism of two graphs is comparing the *unfolding trees*² (UT) rooted at their nodes. First, we formally define what a tree is.

Definition 2.4. A **tree** is a connected acyclic graph. A **rooted tree** is a tree in which one vertex is chosen as the root.

Definition 2.5 (Unfolding Tree). The **unfolding tree** \mathbf{UT}_v^l in graph $G = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ of node $v \in \mathcal{V}$ **up to depth** $l \in \mathbb{N}_0$ is defined as

$$\mathbf{UT}_v^l = \begin{cases} \text{Tree}(\mathbf{x}_v, \emptyset) & \text{if } l = 0 \\ \text{Tree}(\mathbf{x}_v, \mathbf{UT}_{N(v)}^{l-1}) & \text{if } l > 0, \end{cases}$$

where $\text{Tree}(\mathbf{x}_v, \emptyset)$ is a tree consisting of node v with feature \mathbf{x}_v . $\text{Tree}(\mathbf{x}_v, \mathbf{UT}_{N(v)}^{l-1})$ is the tree consisting of the root node v and subtrees $\mathbf{UT}_{N(v)}^{l-1} = \left\{ \left\{ \mathbf{UT}_u^{l-1} \mid u \in N(v) \right\} \right\}$ of depth $l - 1$. The **unfolding tree** of v is defined as $\mathbf{UT}_v = \lim_{l \rightarrow \infty} \mathbf{UT}_v^l$.

Here is an example of the unfolding tree of depth $d = 2$ for node 1 of the graph in Figure 2.2.

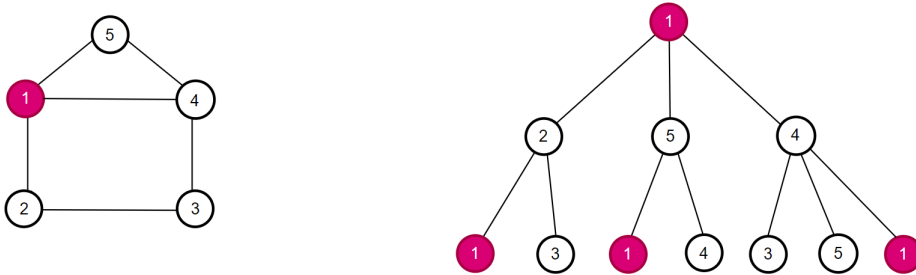


Figure 2.2: The unfolding tree \mathbf{UT}_1^2 for the graph on the left.

[88] and [89] showed that the unfolding tree and 1-WL are equivalent for testing the isomorphism of two graphs, i.e., the colors of the nodes after i iterations of 1-WL are the same if and only if the unfolding trees of depth i are isomorphic.

²The unfolding trees are in some text referred to as *computational graphs* [87].

2.4.3 Higher order versions of the WL test

In [90] the authors devised the more powerful extension k -WL, which colors k -tuples from \mathcal{V}^k instead of single nodes (see [59]).

Definition 2.6 (k -WL Test). Let $N_{s,j} = \{(s_1, \dots, s_{j-1}, r, s_{j+1}, \dots, s_k) \mid r \in \mathcal{V}\}$ be the j -th neighborhood of a k -tuple $s = (s_1, \dots, s_k) \in \mathcal{V}^k$. It is obtained by replacing the j -th component of s by every node from \mathcal{V} . In iteration 0, the algorithm initializes each k -tuple with its *atomic type*, i.e., two k -tuples $s, s' \in \mathcal{V}^k$ get the same color if the assignment $s_i \mapsto s'_i$ induces an isomorphism between the subgraphs induced from the nodes from s and s' , respectively.

For iteration $i > 0$, we define

$$\mathbf{C}_{s,j}^{(i)} = \text{HASH} \left((\mathbf{c}_{s'}^{(i-1)} \mid s' \in N_j(s)) \right),$$

and set

$$\mathbf{c}_s^{(i)} = \text{HASH} \left(\mathbf{c}_s^{(i-1)}, (\mathbf{C}_{s,1}^{(i)}, \dots, \mathbf{C}_{s,k}^{(i)}) \right).$$

Hence, two tuples s and s' with $\mathbf{c}_s^{(i-1)} = \mathbf{c}_{s'}^{(i-1)}$ get different colors in iteration i if there exists $j \leq k$ such that the number of j -neighbors of s and s' , respectively, colored with a certain color is different. The algorithm then proceeds analogously to the 1-WL. By increasing k , the algorithm gets more powerful in terms of distinguishing non-isomorphic graphs, i.e., for each $k \geq 2$, there are non-isomorphic graphs that can be distinguished by the $(k+1)$ -WL but not by the k -WL [61, 4].

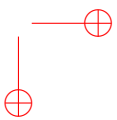
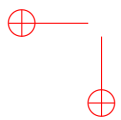
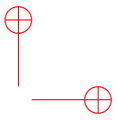
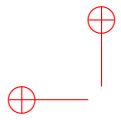
As both are used in GNN literature, we point out the existence of two variants of the same algorithm, which are the folklore k -WL (k -FWL) and the oblivious k -WL (k -OWL). Whenever we mention k -WL, we are referring to k -OWL, for which the following properties hold:

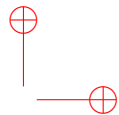
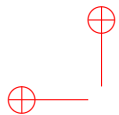
- 1-WL is equivalent in expressive power to 2-WL ([91])
- $(k+1)$ -WL is more expressive than k -WL for any $k \geq 2$ [61].

The existing relation between k -OWL and k -FWL is the following:

$$k\text{-OWL} \equiv (k-1)\text{-FWL},$$

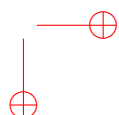
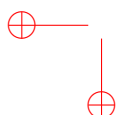
for any $k > 1$ ([92]).

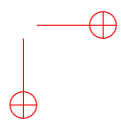
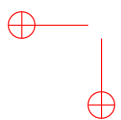
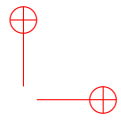
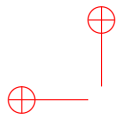




Part II

Going Beyond Standard Graph-Types







Chapter 3

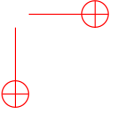
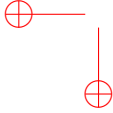
Weisfeiler and Leman go Edge-Attributed and Dynamic

Recently, there has been an increasing interest in studying the theoretical properties of GNNs. Such a trend is partially motivated by the attempt to derive foundational knowledge to design reasonable solutions for many possible applications [85]. A particular focus is on the expressive power of GNNs since the performance of a model also depends on its capability to distinguish different graphs or approximate functions. On the one hand, it has been proven that GNNs are as powerful as the Weisfeiler-Lehman test (1-WL) in their ability to distinguish graphs [26, 59]. Moreover, it has been shown that the equivalence enforced by 1-WL equals the unfolding equivalence [72] [88]. Hence, 1-WL and the unfolding equivalence can be used interchangeably. On the other hand, it has been proven in [93] that the original GNN model can approximate in probability, up to any degree of precision, any measurable function $\tau(\mathbf{G}, v) \rightarrow \mathbb{R}^m$ that respects the unfolding equivalence. Such a result has been recently extended to a large class of GNNs [88] called message-passing GNNs, including most contemporary architectures. Despite the availability of the mentioned results on the expressive power of GNNs, their application is still limited to undirected static graphs with attributes only on nodes. This limitation is particularly restrictive since modern applications usually involve more complex data structures, such as heterogeneous, directed, dynamic graphs and multigraphs. In particular, the ability to process dynamic graphs is progressively gaining significance in many fields such as social network analysis [94], recommender systems [95, 96], traffic forecasting [12] and knowledge graph completion [97, 98]. Several surveys discuss the usage of dynamic graphs in other application domains [99, 100, 101, 102, 103]

Considering the diversity of graph types, it has recently been shown that Static Attributed Undirected Homogeneous Graphs with both attributes on nodes and edges (SAUHG¹) can act as a standard form for graph representation [104]. This means that all the common graph types can be transformed into SAUHGs without losing their encoded information. We collected a detailed description of the transformations from any graph type to SAUHG as additional preliminaries (Appendix B.1).

In this chapter, we aim to study the expressive power of GNNs for arbitrary graph types, analyzing both their separation and approximation capabilities. In particular, we perform two different analyses, one focused on SAUHGs and the other on dynamic graphs. Indeed, despite the possibility of representing a dynamic graph as a SAUHG, the existing

¹Throughout the chapter, we often refer to the SAUHGs simply as "static graphs with attributes also on the edges". This is because all the considered graphs are undirected, homogeneous, and with node attributes. We left the acronym SAUHG, to be consistent with the literature.



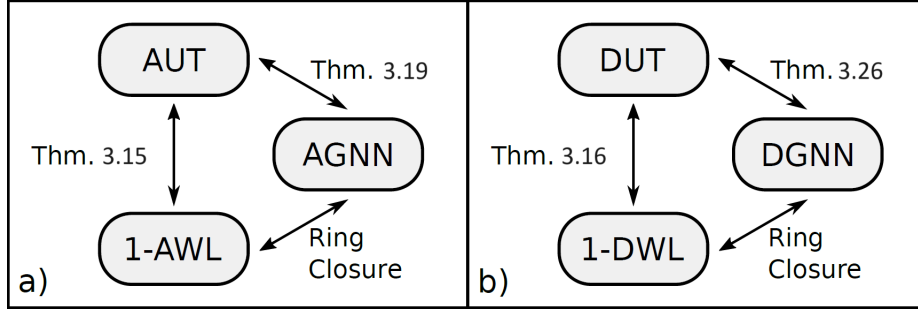


Figure 3.1: a) In Thm. 3.2, we prove the equivalence of the attributed unfolding tree equivalence (AUT) and the attributed 1-WL equivalence (1-AWL) for SAUHGs. Afterward in Thm. 3.5, we show a result on the approximation capability of static GNNs for SAUHGs (AGNN) using the AUT equivalence. b) Analogously to the attributed case, we show similar results for Dynamic GNNs (DGNN) which can be used on temporal graphs.

GNN architectures for dynamic graphs are notably different compared to the static ones. To appreciate the structural difference of architectures tailored for dynamic graphs, we refer the reader to [105, 99, 102], where several models are surveyed. The diagram in Figure 3.1 serves as a visual abstract of our results on separation capabilities. On the left, 3.1a), the results for static graphs with edge-attributes and analogously on the right, 3.1b), the ones for dynamic graphs.

More precisely, the content of the chapter is based on [3] and its outline is as follows.

- We define two new versions of the Weisfeiler-Leman test and of the Unfolding Tree both for SAUHGs and Dynamic graphs.
- We show that the static attributed 1-WL (AWL) and the corresponding Unfolding Tree (AUT) induce the same equivalence on nodes. We prove the equivalence also for the dynamic counterparts (DWL and DUT). Such a result makes it possible to use the WL and UT interchangeably to study the expressiveness of these types of GNNs.
- We show that generic GNN models for dynamic graphs and SAUHGs are capable of approximating, in probability and up to any precision, any measurable function on graphs that respects the *corresponding* 1-WL/unfolding equivalence.
- The result on approximation capability holds for graphs with attributes of reals and unconstrained target functions. Thus, most of the domains used in practical applications are included. Moreover, the proof is constructive, which allows us to deduce information about the GNN architecture that can achieve the desired approximation.
- We validate our theoretical results with an experimental analysis. Our setup shows that sufficiently powerful Dynamic GNNs (DGNNs) can approximate dynamic systems that preserve the unfolding equivalence in contrast to non-universal architectures which lead to poor performances.

3.1 Preliminary definitions

In the following, we formally define the domain of the present investigation: the static edge-attributed graph, and the dynamic graph. Then, we report the architectures specifically designed to handle such graph types, respectively AGNN for edge-attributed graphs and DGNN for dynamic graphs. Their expressiveness and approximation capabilities will be the object of the present investigation.

Definition 3.1 (SAUHG). G is a **Static, (node and edge) Attributed Graph (SAUHG)** if $G = (\mathcal{V}, \mathcal{E}, \alpha, \omega)$, where \mathcal{V} is a finite set of nodes, $\mathcal{E} \subset \{(u, v) \mid \forall u, v \in \mathcal{V}\}$ is a finite set of edges. Node and edge attributes are determined by the mappings $\alpha : \mathcal{V} \rightarrow \mathbb{R}^k$, $\omega : \mathcal{E} \rightarrow \mathbb{R}^{k'}$ that map into the node attribute set \mathbb{R}^k , and edge attribute set $\mathbb{R}^{k'}$ for some $k, k' \in \mathbb{N}$. The domain of SAUHGs will be denoted as \mathcal{G} .

In the following, we will denote with $\alpha_v := \alpha(v)$ and with $\omega_{(u,v)} := \omega((u, v))$. Temporal changes in the graph can be formalized in several ways, here the dynamic graph definition consists of a discrete-time representation.

Definition 3.2 (Dynamic Graph). Let $I = [0, \dots, l] \subsetneq \mathbb{N}_0$ be a sequence of timesteps. Then a **(discrete) dynamic graph** $G = (\mathcal{V}_I, \mathcal{E}_I, \alpha', \omega')$ is a sequence of static graph snapshots, i.e. $G = (G_t)_{t \in I}$, where each G_t is a SAUHG. We define, for the time interval I , the **total set of nodes** of the dynamic graph G as $\mathcal{V}_I := \bigcup_{t \in I} \mathcal{V}_t$. Similarly, the **total set of edges** as $\mathcal{E}_I := \bigcup_{t \in I} \mathcal{E}_t$. Then, the dynamic node attributes α' and edge attributes ω' , are determined respectively by the partial functions $\alpha' : \mathcal{V} \times I \rightarrow A$ and $\omega' : \mathcal{E} \times I \rightarrow B$ defined only for existent nodes and existent edges at each time t :

$$\begin{aligned} \alpha'(v, t) &:= \alpha_t(v) & \text{if } v \in \mathcal{V}_t \\ \omega'((u, v), t) &:= \omega_t((u, v)) & \text{if } (u, v) \in \mathcal{E}_t \end{aligned}$$

where α_t and ω_t are the attributes functions of the graph snapshot, G_t (See Def. 3.1).

In the following, we will denote with $\alpha'_v(t) := \alpha'(v, t)$ and with $\omega'_{(u,v)}(t) := \omega'((u, v), t)$. In our formalization, the dynamic graph is evolving on two levels: its topological structure and its attributes. From a structural perspective, note that the sets of nodes and edges are not static but vary over time, denoted as \mathcal{V}_t and \mathcal{E}_t respectively. Consequently, also the neighborhood of a node $v \in \mathcal{V}_I$ is changing over time, and we will call it $N_t(v)$. If $v \in \mathcal{V}_I$ it means that there exists a time t such that $v \in \mathcal{V}_t$, namely the node exists at some point in the interval I . The same applies to edges. The attributes α_t on nodes and ω_t on edges can change their value over time as well.

Once SAUHGs and dynamic graphs are defined, we can outline the architectures capable of handling those graph types: an Attributed Graph Neural Network (AGNN), which addresses edge attributes, and a discrete Dynamic Graph Neural Network (DGNN), specifically tailored for dynamic graphs.

Given that SAUHGs act as a standard form for all graph types, the ordinary GNN architecture will be extended to take edge attributes into account. This can be done by

including the edge attributes in the general message passing GNN framework (See Preliminaries 2.3), as follows.

Definition 3.3 (AGNN). For a SAUHG $G = (\mathcal{V}, \mathcal{E}, \alpha, \omega)$ let $u, v \in \mathcal{V}$. We initialize the feature of node v as $\mathbf{h}_v^{(0)} = \alpha_v$ for all $v \in \mathcal{V}$. The AGNN propagation scheme for iteration $i > 0$ is defined as

$$\mathbf{h}_v^{(i)} = \text{COMBINE}^{(i)} \left(\mathbf{h}_v^{(i-1)}, \text{AGGREGATE}^{(i)} \left(\{\{\mathbf{h}_u^{(i-1)}\}_{u \in \mathcal{N}(v)}, \{\{\omega_{(u,v)}\}_{u \in \mathcal{N}(v)}\}\} \right) \right)$$

The output for a node-specific learning problem after the last iteration L is given by

$$\mathbf{z}_v = \text{READOUT} \left(\mathbf{h}_v^{(L)} \right),$$

using a selected aggregation scheme and a suitable READOUT function. Similarly, the output for a graph-specific learning problem is determined by

$$\mathbf{z} = \text{READOUT} \left(\{\{\mathbf{h}_v^{(L)} \mid v \in \mathcal{V}\}\} \right).$$

In this particular model, the attributes on the edges remain constant. Therefore they play a role in the initialization of the node features but they do not change during the learning phase. Moreover, the feature of the edge (u, v) is aggregated independently from the features of the nodes u and v .

For the dynamic case, we chose a widely used dynamic GNN model from [99]. This particular discrete dynamic graph neural network (DGNN) uses a recurrent network to encode the temporal evolution and a standard GNN to encode each graph snapshot. Here, we substitute the standard GNN with the previously defined AGNN.

Definition 3.4 (DGNN). Given a discrete dynamic graph $G = (G_t)_{t \in I}$, a **discrete DGNN** using a continuously differentiable recursive function f for temporal modelling can be expressed as:

$$\begin{aligned} \mathbf{q}_{v_1}(0), \dots, \mathbf{q}_{v_n}(0) &= \mathbf{h}_{v_1}(0), \dots, \mathbf{h}_{v_n}(0) := \text{AGNN}(G_0) \\ \mathbf{h}_{v_1}(t), \dots, \mathbf{h}_{v_n}(t) &:= \text{AGNN}(G_t) \quad \forall t \geq 0 \\ \mathbf{q}_{v_i}(t) &:= f(\mathbf{q}_{v_i}(t-1), \mathbf{h}_{v_i}(t)) \quad \forall v_i \in \mathcal{V} \end{aligned} \quad (3.1)$$

where $\mathbf{h}_{v_i}(t) \in \mathbb{R}^r$ is the hidden representation of node v_i at time t of dimension r and $\mathbf{q}_{v_i}(t) \in \mathbb{R}^s$ is an s -dimensional hidden representation of node v_i produced by f , and $f : \mathbb{R}^s \times \mathbb{R}^r \rightarrow \mathbb{R}^s$ is a neural architecture for temporal modeling (in the methods surveyed in [99], f is almost always an RNN or an LSTM). Depending on the task, if graph or node focused, a suitable $\text{READOUT}_{\text{dyn}}$ function will be used to get the final output.

Once we have defined the architectures dealing with the domains of interest we aim to extensively study their expressive capabilities. The study of expressive power can be approached from two perspectives: one focuses on distinguishing graphs, and the other on approximating functions. These two perspectives are closely related. For the former, we define new versions of the Weisfeiler-Leman test characterizing the expressivity of AGNN

and DGNN. Similarly, we define the corresponding unfolding trees and demonstrate that they induce the same equivalence as the WL tests do. Finally, we prove that all the functions preserving this equivalence can be approximated by the AGNN and DGNN architectures.

3.2 Edge-Attributed and Dynamic Extensions of 1-WL test and Unfolding Tree

In this section, the extensions of 1-WL and Unfolding Tree are collected. From [72], it is known that for static node-attributed graphs, both the unfolding tree and the Weisfeiler-Leman approach are equivalent for testing the isomorphism of two graphs. Thus, we first extend the notion of unfolding trees and the Weisfeiler-Leman test to an edge-attributed version.

Definition 3.5 (Edge-Attributed 1-WL test). Let Hash be a bijective function that codes every possible node attribute with a color from a color set Σ and $G = (\mathcal{V}, \mathcal{E}, \alpha, \omega)$. The **edge-attributed 1-WL (1-AWL) test** is defined recursively through the following.

- At iteration $i = 0$, the color is set to the hashed node attribute:

$$\mathbf{c}_v^{(0)} = \text{Hash}(\alpha_v)$$

- At iteration $i > 0$, the Hash function is extended to the edge attributes:

$$\mathbf{c}_v^{(i)} = \text{Hash}\left(\left(\mathbf{c}_v^{(i-1)}, \{\{\mathbf{c}_u^{(i-1)}\}_{u \in \mathcal{N}(v)}, \{\{\omega_{(u,v)}\}_{u \in \mathcal{N}(v)}\}\right)\right)$$

In the following, the 1-WL equivalence of graphs and nodes is extended by using the edge-attributed version of the 1-WL test.

Definition 3.6 (Attributed 1-WL equivalence). Two nodes u, v are attributed WL-equivalent, noted by $u \sim_{AWL} v$, if and only if $\mathbf{c}_u = \mathbf{c}_v$.

Analogously, let $G_1 = (\mathcal{V}_1, \mathcal{E}_1, \alpha_1, \omega_1)$ and $G_2 = (\mathcal{V}_2, \mathcal{E}_2, \alpha_2, \omega_2)$ be two SAUHG. Then, $G_1 \sim_{AWL} G_2$, if and only if for all nodes $v_1 \in \mathcal{V}_1$ there exists a corresponding node $v_2 \in \mathcal{V}_2$ such that $v_1 \sim_{AWL} v_2$.

Definition 3.7 (Edge-Attributed Unfolding Tree). The **attributed unfolding tree** \mathbf{T}_v^d in graph $G = (\mathcal{V}, \mathcal{E}, \alpha, \omega)$ of node $v \in \mathcal{V}$ **up to depth** $d \in \mathbb{N}_0$ is defined as

$$\mathbf{T}_v^d = \begin{cases} \text{Tree}(\alpha_v), & \text{if } d = 0 \\ \text{Tree}\left(\alpha_v, \{\{\omega_{(u,v)}, \mathbf{T}_u^{d-1}\}_{u \in \mathcal{N}(v)}\}\right) & \text{if } d > 0, \end{cases}$$

where $\text{Tree}(\alpha_v)$ is a tree constituted of node v with attribute α_v , and the expression $\text{Tree}\left(\alpha_v, \{\{\omega_{(u,v)}, \mathbf{T}_u^{d-1}\}_{u \in \mathcal{N}(v)}\}\right)$ denotes the tree of depth d consisting of the root node v connected to trees of depth $d - 1$ rooted at each neighbor u of v , namely \mathbf{T}_u^{d-1} . The edges connecting the root v to the trees are weighted by the corresponding edge attribute $\omega_{(u,v)}$, for every neighbor $u \in \mathcal{N}(v)$. We call **attributed unfolding tree** of v , denoted by \mathbf{T}_v , the tree determined by $\mathbf{T}_v = \lim_{d \rightarrow \infty} \mathbf{T}_v^d$.

Definition 3.8 (Attributed Unfolding Equivalence). Let $G_1 = (\mathcal{V}_1, \mathcal{E}_1, \alpha_1, \omega_1)$ and $G_2 = (\mathcal{V}_2, \mathcal{E}_2, \alpha_2, \omega_2)$ be two SAUHGs. Then G_1 and G_2 are **attributed unfolding tree equivalent**, noted by $G_1 \sim_{AUT} G_2$, iff $\{\{\mathbf{T}_u \mid u \in \mathcal{V}_1\}\} = \{\{\mathbf{T}_v \mid v \in \mathcal{V}_2\}\}$. Analogously, two nodes $u \in \mathcal{V}_1, v \in \mathcal{V}_2$ are unfolding tree equivalent, noted by $u \sim_{AUT} v$ if and only if $\mathbf{T}_u = \mathbf{T}_v$.

Now we can define the dynamic versions, extending the attributed ones to a time-dependent approach. We employ SAUHGs as a standard form for the static graph at each timestep, as they represent the most general case of static graphs.

Definition 3.9 (Dynamic 1-WL test). Let $G = (\mathcal{V}_I, \mathcal{E}_I, \alpha', \omega')$ be a dynamic graph. Let Hash_t^0 be an injective function encoding every node attribute at time t with a color from a color set Σ . The **dynamic 1-WL test (1-DWL)** generates a dynamic color \mathbf{c}_v for each node $v \in \mathcal{V}$, that is a sequence of attributed colors $\mathbf{c}_v(t)$ over the time interval I , following the iterative procedure described below.

- At iteration $i = 0$ and time t , the color of node v is set to the hashed node attribute or, whether v doesn't exist, to a fixed color \mathbf{c}^\perp :

$$\mathbf{c}_v^{(0)}(t) = \begin{cases} \text{Hash}_t^0(\alpha'_v(t)) & \text{if } v \in \mathcal{V}_t, \\ \mathbf{c}^\perp & \text{otherwise.} \end{cases}$$

- Then, the aggregation mechanism at time t is defined by the injective function Hash_t , for $i > 0$:

$$\mathbf{c}_v^{(i)}(t) = \text{Hash}_t \left((\mathbf{c}_v^{(i-1)}(t), \{\{\mathbf{c}_u^{(i-1)}(t)\}_{u \in N_v(t)}, \{\{\omega_{(u,v)}(t)\}_{u \in N_v(t)}\}) \right)$$

Note that if a node doesn't exist at time t , its color is $\mathbf{c}_v^{(i)}(t) = \mathbf{c}^\perp$ for every $i \geq 0$ because its neighborhood is empty. This implies that non-existent nodes cannot influence the neighborhood aggregation of other nodes, or equivalently, that they cannot propagate through the graph. Note that the edge attributes contribute from the iteration $i = 1$ in discriminating more the colors of the nodes.

Definition 3.10 (Dynamic 1-WL equivalence). Two nodes $u, v \in \mathcal{V}$ in a dynamic graph G are said to be **dynamic WL equivalent**, noted by $u \sim_{DWL} v$, if their colors resulting from the WL test are pairwise equal for each timestep. Analogously, let G and G' be two dynamic graphs. Then $G \sim_{DWL} G'$, iff $\{\{\mathbf{c}_u \mid u \in \mathcal{V}\}\} = \{\{\mathbf{c}_v \mid v \in \mathcal{V}'\}\}$.

In Figure 3.2 we show two iterations of DWL test.

Remark 3.1. Note that the dynamic WL test is not just a multi-dimensional extension of the attributed one as it may seem, because the graph snapshots are not independent. This means that even if all the graph snapshots are pairwise equivalent via AWL, the dynamic graphs can be DWL non-equivalent. For example, look at the dynamic graphs G and G' in Figure 3.3. If a node doesn't exist at a certain timestep, it is represented with a dotted line. At each timestep $t = 1, 2, 3$, $G_t \sim_{AWL} G'_t$ but $G \not\sim_{DWL} G'$. Indeed,

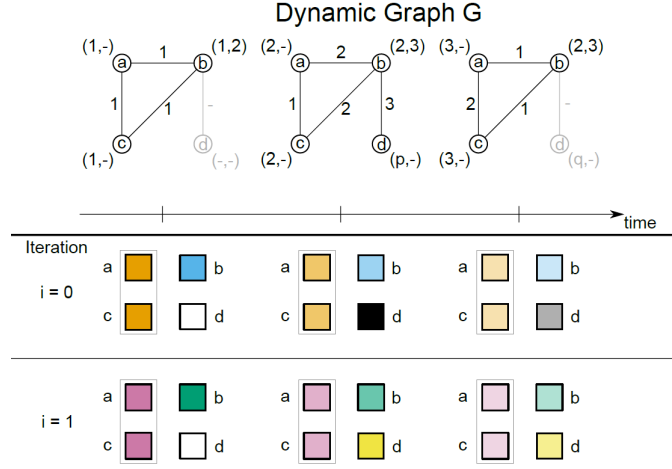


Figure 3.2: DWL test on the graph G composed of three static graph snapshots. Taken from [3].

$G_t \sim_{AWL} G'_t$ if and only if you find a bijection between the nodes of the graphs such that $v \sim_{AWL} v'$ for $v \in \mathcal{V}_t$ and $v' \in \mathcal{V}'_t$. The bijection is not necessarily the same for every timestep (once v is equivalent to v' , once to u'). Instead, to guarantee the dynamic equivalence, there must exist a bijection on the nodes such that, $v \sim_{DWL} v'$ for $v \in \mathcal{V}_I$ and $v' \in \mathcal{V}'_I$. Here, the nodes in the *total* sets \mathcal{V}_I and \mathcal{V}'_I , are coupled once for all the timestep t , that is $v \sim_{DWL} v'$ iff for all t , $v \sim_{AWL} v'$. Equivalently, we compare the sequence of colors for each node, and two sequences are equal, if they are equal for all the timestep, also the *order* of the colors is taken into account. We consider the *entity* of the nodes as permanent (the nodes in the total set), their state (existent/non-existent) is time-dependent and encoded in their colors.

Definition 3.11 (Dynamic Unfolding Tree). Let $G = (\mathcal{V}_I, \mathcal{E}_I, \alpha', \omega')$ be a dynamic graph. The **dynamic unfolding tree** \mathcal{T}_v^d of node $v \in \mathcal{V}_I$ up to depth $d \in \mathbb{N}_0$ is a sequence of attributed unfolding trees $\mathcal{T}_v^d := (\mathbf{T}_v^d(t))_{t \in I}$. If $v \in \mathcal{V}_t$, then $\mathbf{T}_v^d(t)$ is defined in the following:

$$\mathbf{T}_v^d(t) := \begin{cases} Tree(\alpha'_v(t)), & \text{if } d = 0 \\ Tree(\alpha'_v(t), \{\{(\omega'_{u,v}(t), \mathbf{T}_u^{d-1}(t))\}_{u \in N_v(t)}\}) & \text{if } d > 0, \end{cases}$$

where, at each time t , the tree corresponds to the attributed unfolding tree of Def. 3.7. If the node v doesn't exist at time t , namely $v \notin \mathcal{V}_t$, then $\mathbf{T}_v^d(t) := \mathbf{T}^\perp$ for every d .

Definition 3.12 (Dynamic Unfolding Equivalence). Two nodes $u, v \in \mathcal{V}$ are said to be **dynamic unfolding equivalent** $u \sim_{DUT} v$ iff $\mathcal{T}_u = \mathcal{T}_v$, that is $\mathbf{T}_u(t) = \mathbf{T}_v(t)$ for every timestep t . Analogously, two dynamic graphs G, G' are said to be **dynamic unfolding equivalent** $G \sim_{DUT} G'$, iff $\{\{\mathcal{T}_u \mid u \in \mathcal{V}\}\} = \{\{\mathcal{T}_v \mid v \in \mathcal{V}'\}\}$.

Finally, we are going to prove that the attributed/dynamic version of the unfolding tree and the WL test, induce the same equivalence on nodes, based on the following

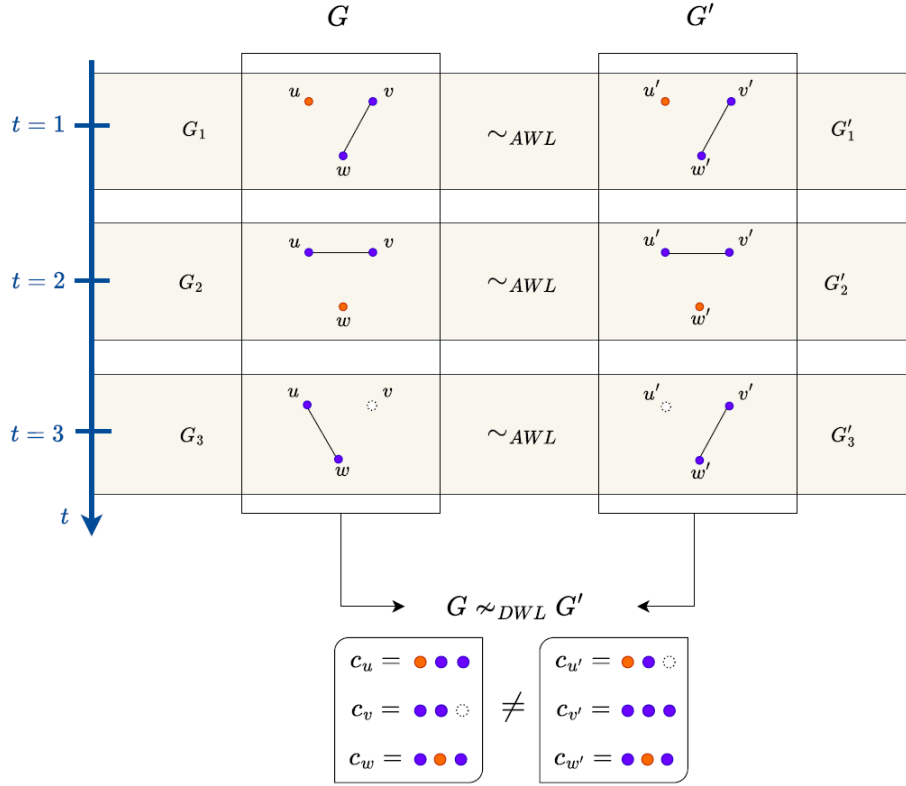


Figure 3.3: Graphs G and G' that are AWL equivalent for every snapshot but not DWL equivalent.

helping lemma. In [93], it has been shown that the unfolding trees of infinite depth are not necessary to consider for this equivalence. Instead, the order of the graph is sufficient for the depth of the unfolding trees, which is finite since the graph is bounded. The following lemma determines the equivalence between the attributed unfolding trees of two nodes and their colors resulting from the attributed 1-WL test.

Lemma 3.1. *Let $G = (\mathcal{V}, \mathcal{E}, \alpha, \omega)$ be a SAUHG and nodes $u, v \in \mathcal{V}$. Then it holds that*

$$\forall d \in \mathbb{N}_0 : \mathbf{T}_u^d = \mathbf{T}_v^d \iff \mathbf{c}_u^{(d)} = \mathbf{c}_v^{(d)}.$$

A complete proof can be found in Appendix B.2.1.

Directly from Lemma 3.1, we can formalize that the unfolding tree equivalence and the attributed 1-WL equivalence of two nodes are indeed the same, both for edge-attributed graphs and dynamic graphs.

Theorem 3.2 (AWL=AUT). *Let $G = (\mathcal{V}, \mathcal{E}, \alpha, \omega)$ be a SAUHG. Then, it holds*

$$\forall u, v \in \mathcal{V} : u \sim_{AUT} v \iff u \sim_{AWL} v.$$

The result is a direct consequence of Lemma 3.1.

Theorem 3.3 (DWL=DUT). *Let G be a discrete dynamic graph where \mathcal{V} is the total set of nodes. Then, it holds*

$$\forall u, v \in \mathcal{V} : u \sim_{DUT} v \iff u \sim_{DWL} v.$$

Proof. Two nodes are dynamic unfolding/WL equivalent iff they are attributed unfolding/WL equivalent at each timestep, that is

$$u \sim_{DUT} v \iff u \sim_{AUT} v \quad \forall t \in I$$

and

$$u \sim_{DWL} v \iff u \sim_{AWL} v \quad \forall t \in I$$

Then, as a consequence of Theorem 3.2, we can conclude that the equivalence induced by DUT and DWL is the same.

The theorem covers the case of non-existent nodes. Indeed, two non-existent nodes are AWL-equivalent at time t if they have the same color \mathbf{c}^\perp and they are AUT-equivalent if they have the same tree \mathbf{T}^\perp . □

Thanks to these results, the unfolding equivalence and the WL equivalence will be used interchangeably in the next chapters.

3.3 Universal Approximation Capabilities for AGNN and DGNN

In this section, the results from Section 3.2 are brought together in the formulation of a universal approximation theorem for GNNs working on SAUHG and dynamic graphs. Analogously to [93] and [88], we are going to prove that AGNNs and DGNNs can approximate all the functions that preserve the attributed or dynamic unfolding equivalence.

3.3.1 GNNs for Edge-Attributed Static Graphs

First of all, it is necessary to define the family of functions preserving the attributed unfolding equivalence. A function on nodes preserves the attributed unfolding equivalence if the output of the function is equal when two nodes are attributed unfolding equivalent.

Definition 3.13 (Functions preserving the AUT equivalence). Let $G = (\mathcal{V}, \mathcal{E}, \alpha, \omega)$ be a SAUHG and $u, v \in \mathcal{V}$ two nodes. Then a function $f : G \times \mathcal{V} \rightarrow \mathbb{R}^m$ is said to **preserve the attributed unfolding equivalence** if

$$v \sim_{AUT} u \Rightarrow f(G, v) = f(G, u)$$

All the functions that preserve the attributed unfolding equivalence on the domain \mathcal{G} of SAUHG, are collected in the set $\mathcal{F}(\mathcal{G})$.

With an argument analogous to [93], we show that there exists a relation between the functions preserving the unfolding equivalence and the attributed unfolding, as follows.

Proposition 3.4 (Functions of attributed unfolding trees). *A function f belongs to $\mathcal{F}(\mathcal{G})$ if and only if there exists a function κ defined on trees such that for any graph $G \in \mathcal{G}$ it holds $f(G, v) = \kappa(\mathbf{T}_v)$, for any node $v \in G$.*

The proof works analogously to its unattributed version presented in [93] and can be found in Appendix B.2.2.

We can finally state the universal approximation capability of AGNNs on SAUHG. We consider only finite graphs.

Theorem 3.5 (Universal Approximation Theorem by AGNN). *Let \mathcal{G} be the domain of bounded SAUHGs with the maximal number of nodes $N = \max_{G \in \mathcal{G}} |G|$. For any measurable function $f \in \mathcal{F}(\mathcal{G})$, any norm $\|\cdot\|$ on \mathbb{R}^m , any probability measure P on \mathcal{G} , and for any positive reals ϵ, λ , the following holds.*

There exists an AGNN, defined by the continuously differentiable functions COMBINE, AGGREGATE and READOUT, with embedding vector of dimension $r = 1$, i.e. $\mathbf{h}_v \in \mathbb{R}$, such that the function φ realized by the AGNN after $2N - 1$ steps, satisfies the condition

$$P(\|f(G, v) - \varphi(G, v)\| \leq \epsilon) \geq 1 - \lambda, \quad \forall G \in \mathcal{G}, \forall v \in \mathcal{V}.$$

The corresponding proof can be found in the Appendix B.2.4.

Theorem 3.5 essentially states that given a function f that preserves the AWL equivalence, there exists an AGNN that approximates it in probability.

Remark 3.2. Note that the theorem has few requirements on the components defining the AGNN. That is, COMBINE, AGGREGATE, and READOUT have no other constraints than to be continuously differentiable. They are completely generic (differentiable) functions. This situation does not correspond to practical cases where the AGNN adopts particular architectures, and those functions are neural networks, or more generally, parametric models. For example, they can be made of layers of sum, max, average, etc. Thus, it is of great interest to clarify whether the theorem still holds when the components of the AGNN are parametric models.

Having established the *theoretical* approximation capability of AGNNs, we now look for an approximation theorem that more closely aligns with the AGNNs implemented in *practice*.

Following [93], it turned out that the class of networks that are sufficiently general to be able to approximate any function preserving the unfolding equivalence is the class of AGNNs with *universal components*.

Definition 3.14 (Universal Components). A class of AGNN models is said to have *universal components* if, for every $\epsilon > 0$ and any continuous target functions $\overline{\text{COMBINE}}$, $\overline{\text{AGGREGATE}}$, $\overline{\text{READOUT}}$, there exists an AGNN composed by parametric models COMBINE_θ , AGGREGATE_θ , READOUT_θ and parameters θ such that

$$\begin{aligned} \|\overline{\text{COMBINE}}(\mathbf{h}, \overline{\text{AGGREGATE}}(\{\mathbf{h}_1, \dots, \mathbf{h}_n\})) - \text{COMBINE}_\theta(\mathbf{h}, \text{AGGREGATE}_\theta(\{\mathbf{h}_1, \dots, \mathbf{h}_n\}))\|_\infty &\leq \epsilon, \\ \|\overline{\text{READOUT}}(\mathbf{q}) - \text{READOUT}_\theta(\mathbf{q})\|_\infty &\leq \epsilon, \end{aligned}$$

holds, for any vectors $\mathbf{h}, \mathbf{h}_1, \dots, \mathbf{h}_n \in \mathbb{R}^r$, $\mathbf{q} \in \mathbb{R}^s$.

We call $\mathcal{Q}_{\mathcal{A}}$, the class of AGNNs with universal components. The following result shows that Theorem 3.5 holds for AGNNs with universal components.

Theorem 3.6 (Approximation by Neural Networks). *Let assume that the hypotheses of Theorem 3.5 are fulfilled. Then, there exists a parameter set θ and some functions $\text{COMBINE}_{\theta}^{(i)}$, $\text{AGGREGATE}_{\theta}^{(i)}$, READOUT_{θ} , implemented by Neural Networks in $\mathcal{Q}_{\mathcal{A}}$, such that Theorem 3.5 holds.*

Namely, if the employed components of the AGNN are universal approximators, then the AGNN can approximate any function preserving the AUT equivalence. The proof is identical to the one contained in [93]; to give a hint on the methodology, we refer to the more complex proof of the analogous Theorem 3.9 for DGNNs.

3.3.2 GNNs for Dynamic Graphs

In this section, the analysis performed for static graphs with edge attributes is extended to cover the dynamic case. First of all, we define the dynamic system, which is a function in the dynamic graphs domain.

Definition 3.15 (Dynamic System). Let \mathcal{D} be the domain of dynamic graphs and let $\mathcal{V}_I = \bigcup_{t \in I} \mathcal{V}_t$. A **dynamic system** is a function $\text{dyn} : \mathcal{D}^* := I \times \mathcal{D} \times \mathcal{V}_I \rightarrow \mathbb{R}^m$ defined as

$$\text{dyn}(t, G, v) := g(x_v(t)) \quad \forall v \in \mathcal{V}_t. \quad (3.2)$$

Here, $g : \mathbb{R}^r \rightarrow \mathbb{R}^m$ is an output function while $x_v(t)$ represents the *state function* and is determined by

$$x_v(t) = \begin{cases} a(t, G, v) & \text{if } t = 0 \\ \text{up}(x_v(t-1), a(t, G, v)) & \text{if } t > 0, \end{cases}$$

where $a : I \times \mathcal{D} \times \mathcal{V}_I \rightarrow \mathbb{R}^r$ is a function that processes the graph snapshot G_t at time t and provides an r -dimensional internal state representation for each node $v \in \mathcal{V}_t$. Finally, $\text{up} : \mathbb{R}^r \times \mathbb{R}^r \rightarrow \mathbb{R}^r$ is a recursive function, that is called *state update function*.

Definition 3.16. A dynamic system $\text{dyn}(\cdot, \cdot, \cdot)$ **preserves the dynamic unfolding equivalence** on \mathcal{D}^* if and only if for any input graph sequences $G, G' \in \mathcal{D}$ in the time interval I , and two nodes $v \in \mathcal{V}_I$, $u \in \mathcal{V}'_I$ it holds

$$v \sim_{DUT} u \implies \text{dyn}(t, G, v) = \text{dyn}(t, G', u) \quad \forall t \in I.$$

The class of dynamic systems that preserve the unfolding equivalence on \mathcal{D}^* will be denoted by $\mathcal{F}(\mathcal{D}^*)$. A characterization of $\mathcal{F}(\mathcal{D}^*)$ is given by the following result (similar to [93]).

Proposition 3.7 (Functions of dynamic unfolding trees). *A dynamic system $\text{dyn}(\cdot, \cdot, \cdot)$ belongs to $\mathcal{F}(\mathcal{D}^*)$ if and only if there exists a function κ defined on attributed trees such that for all the triplets $(t, G, v) \in \mathcal{D}^*$ it holds*

$$\text{dyn}(t, G, v) = \kappa\left(\left(\mathbf{T}_v(i)\right)_{i \in [t]}\right).$$

The proof can be found in Appendix B.2.3.

Finally, the universal approximation theorem for static GNNs can be extended to the discrete dynamic graph neural networks, as follows.

Theorem 3.8 (Universal Approximation Theorem by DGNN). *Let $G = (G_t)_{t \in I}$ be a discrete dynamic graph in the domain \mathcal{D} and $N = \max_{G \in \mathcal{D}} |G|$ be the maximal number of nodes in the domain. Let $\text{dyn}(t, G, v) \in \mathcal{F}(\mathcal{D})$ be any measurable dynamical system preserving the unfolding equivalence, $\|\cdot\|$ be a norm on \mathbb{R}^m , P be any probability measure on \mathcal{D}^* and ϵ, λ be any positive real numbers. Then, there exists a DGNN such that the function φ realized by this model satisfies*

$$P(\|\text{dyn}(t, G, v) - \varphi(t, G, v)\| \leq \epsilon) \geq 1 - \lambda \quad \forall (t, v, G) \in \mathcal{D}^*.$$

The proof can be found in the Appendix B.2.5.

Theorem 3.8 states that given a dynamical system that preserves the DWL equivalence, there exists a DGNN that approximates it. In particular, the DGNN that realizes the theorem is the composition of a recursive function that is continuously differentiable, with AGNNs that satisfy the hypothesis of Theorem 3.5. Moreover, to reach the desired approximation, it is sufficient to have real numbers as embedding "vectors" for the AGNNs and the recurrent model. As observed in Remark 3.2, we notice that the components of the DGNN need solely to be continuously differentiable to make the DGNN a universal approximator modulo the DUT equivalence. What happens if these components are approximated by neural networks or in general, by parametric models? Does the theorem still hold? Analogously to the attributed case, we define the class of models with universal components, for which the theorem holds.

Definition 3.17. A class of discrete DGNN models is said to have *universal components* if the employed AGNNs lay in the class $\mathcal{Q}_{\mathcal{A}}$ (see Def. 3.14) and the employed recurrent model f_{θ} is designed such that for every $\epsilon_1 > 0$ and any continuously differentiable target function \bar{f} , it holds

$$\|\bar{f}(\mathbf{q}, \mathbf{h}) - f_{\theta}(\mathbf{q}, \mathbf{h})\|_{\infty} \leq \epsilon_1,$$

for any input vectors $\mathbf{h} \in \mathbb{R}^r$, $\mathbf{q} \in \mathbb{R}^s$.

Moreover, the output of the DGNN, given by the parametric function $\text{READOUT}_{\text{dyn}, \theta}$, must satisfy for every $\epsilon_2 > 0$:

$$\|\overline{\text{READOUT}}_{\text{dyn}}(\mathbf{q}^*) - \text{READOUT}_{\text{dyn}, \theta}(\mathbf{q}^*)\|_{\infty} \leq \epsilon_2,$$

for every continuously differentiable target function $\overline{\text{READOUT}}_{\text{dyn}}$ and every input vector $\mathbf{q}^* \in \mathbb{R}^s$.

We denote with $\mathcal{Q}_{\mathcal{D}}$, the class of discrete DGNNs with universal components. An instance of this class is a DGNN composed of neural networks whose universality has been established, e.g. MLP [27] and RNN [106]. The following result shows that Theorem 3.8 holds for discrete DGNNs with universal components.

Theorem 3.9 (Approximation by Neural Networks). *If the hypotheses of Theorem 3.8 are fulfilled, then there exists a parameter set θ , and functions \bar{f}_{θ} , $\text{READOUT}_{\text{dyn}, \theta}$ implemented by neural networks in $\mathcal{Q}_{\mathcal{D}}$, such that Theorem 3.8 holds.*

The proof can be found in the Appendix B.2.6.

Discussion. The following remarks may further help to understand the results proven in the previous paragraphs:

- Theorem 3.5 suggests an alternative approach to process several graph domains with a unique and universal AGNN model. Indeed almost all the graphs - including hypergraphs, multigraphs, directed graphs, ... - can be transformed to SAUHGs with node and edge attributes [104].
- The proofs of Theorems 3.5 and 3.8 are based on space partitioning reasoning, that is a constructive procedure. Differently from the technique based on the Stone-Weierstrass theorem [30], which is existential in nature, such an approach allows us to deduce information about the details of the networks that reach the desired approximation. The theorems point out that the approximation can be obtained with a hidden dimension $r = 1$, both in AGNNs and DGNNs. At the same time, the required state dimension for the Recurrent Neural Network of DGNNs is still equal to 1. Such a result may appear surprising, but the proofs show that GNNs can encode unfolding trees with a single real number.
- Moreover, Theorems 3.5 and 3.8 specify that GNNs can obtain the approximation with $2N - 1$ layers. We may incorrectly presume that the maximum number of layers required to reach the desired approximation depends on the diameter $diam(G)$ of the graph, which can be smaller than the number of nodes N since the information in a GNN can *flow* from one node to another in $diam(G)$ iterations. However, $diam(G)$ layers are not always sufficient to distinguish all the nodes of a graph. In fact, it has been proven that $N - 1$ is a lower bound on the number of iterations that the 1-WL algorithm has to carry out to be able to distinguish any pairs of 1-WL distinguishable graphs [107], and $2N - 1$ is a lower bound for 1-WL algorithm to distinguish pairs of nodes in two different graphs [72]. So overall, $2N - 1$ is also the lower bound for the GNN computation time to approximate any function for either graph-focused or node-focused tasks (see [88] for a detailed discussion).
- Theorems 3.5 and 3.8 specify that the approximation is modulo the unfolding equivalence (or modulo the WL equivalence). We observe that in the dynamic case, only a part of the architecture limits the set of functions that can be computed by the DGNN. Indeed, the dynamic GNN contains two modules: the AGNN, producing an embedding of the graph at each time step, and the Recurrent Neural Network, which processes the sequence of the graph snapshots. Intuitively, the Recurrent Neural Network does not affect the equivalence, since Recurrent Neural Networks can be universal approximators [106] and implement any function of the sequence without introducing other constraints beyond those already introduced by the AGNN.
- Theorem 3.8 does not hold for every Dynamic GNN, as we consider a discrete recurrent model working on sequences of graph snapshots (also known as Stacked

DGNN). Nevertheless, several DGNNs of this kind are listed in [105], such as GCRN-M1 [108], RgCNN [109], PATCHY-SAN [110], DyGGNN [111], and others.

3.4 Experimental Validation

In this Section, we validate our theoretical findings with an experimental study. For this purpose, we carry out two sets of experiments described as follows:

- E1.** We show that a DGNN with universal components can approximate a function $F_{DWL} : \mathcal{D} \rightarrow \mathbb{N}$ that models the 1-DWL test. The function F_{DWL} assigns to the input graph a target label representing its class of 1-DWL equivalence;
- E2.** In the same approximation task, we compare DGNNs with different GNN modules from the literature to show how the universality of the components affects the approximation capability.

We focus on the ability of the DGNN to approximate F_{DWL} , so only training performances are considered, i.e., we do not investigate the generalization capabilities over a test set. Since the 1-DWL test provides the finest partition of graphs reachable by a DGNN, the mentioned tasks experimentally evaluate the expressive power of DGNNs.

Dataset. The dataset consists of dynamic graphs, i.e., vectors of static graph snapshots of fixed length T . Each static snapshot is one of the graphs in Fig. 3.4. Since the dataset is composed of all the possible combinations of the four graphs, it contains 4^T dynamic graphs. Given that the graphs in Fig. 3.4 are pairwise 1-WL equivalent (a) is 1-WL equivalent to b) and c) is 1-WL equivalent to d)), the number of classes is 2^T , with $\frac{4^T}{2^T} = 2^T$ graphs in each class. For each dynamic graph, the target is the corresponding 1-DWL output, represented as a natural number. For training purposes, the targets are normalized between 0 and 1 and uniformly spaced in the interval $[0, 1]$. Therefore, the distance between each class label is $d = 1/2^T$. A dynamic graph G with target y_G will be said to be correctly classified if, given $\text{out} = \text{DGNN}(G)$, we have $|\text{out} - y_G| < d/2$.

Experimental setup.

E1 For the first set of experiments, the Dynamic Graph Neural Network is composed of two modules: A Graph Isomorphism Network (GIN) [26] and a Recurrent Neural Network (RNN), which implement the static GNN and the temporal Network f of Eq. (3.1), respectively. Since it has been proven that the GIN is a universal architecture [26] and the RNNs are universal approximators for dynamical systems on vector sequences [106], the architecture used in the experiments fits the hypothesis of Theorem 3.8. Thus, it can approximate any dynamical system on the temporal graph domain.

The model hyperparameters for the experiments are set as follows. The GIN includes

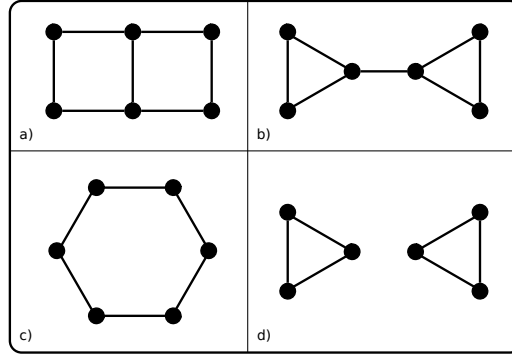


Figure 3.4: These four static graphs are used as components to generate the synthetic dataset. Graphs a) and b) are equivalent under the static 1-WL test; the same holds for c) and d).

$n_{\max} = 6$ layers². The MLP in the GIN network contains one hidden layer with a hyperbolic tangent activation function and batch normalization. Hidden layers of different sizes, i.e., $h_{\text{gin}} \in \{1, 4, 8\}$, have been tested. For sake of simplicity, the output network has one hidden layer with the same number of neurons as the MLP in the GIN. Furthermore, $h_{\text{rnn}} = 8$ is the size of the hidden state of the RNN.

E2 In the other set of experiments, we test DGNNs composed by different GNN static modules and an RNN module (analogously to **E1**). In particular, we compare DGNNs with the GNN module taken from the following list:

- GIN as mentioned before;
- Graph Convolutional Network (GCN) [22];
- GNN presented in [15] (see also [59]) where the aggregation function is the *sum* of the hidden features of the neighbours; it will be called `gconv_add`;
- GNN presented in [15] with *mean* of the hidden features of the neighbours as aggregation function, called `gconv_mean` here;
- GAT [24].

Here, the used hyperparameters are hidden dimension is $h = 8$, the number of layers $L = 4$, and the time length $L = T = 5$.

In both the experimental cases, the model is trained over 300 epochs using the Adam optimizer with a learning rate of $\lambda = 10^{-3}$. Each configuration is evaluated over 10 runs. The overall training is then performed on an Intel(R) Core(TM) i7-9800X processor running at 3.80GHz using 31GB of RAM and a GeForce GTX 1080 Ti GPU unit. The code used to run the experiments can be found at <https://github.com/AleDinve/dyn-gnn>.

²As investigated in the discussion paragraph 3.3.2, for graph-focused tasks, it is sufficient to perform the message-passing convolution for several times equal to the maximum number of nodes over the graphs in the dataset domain.

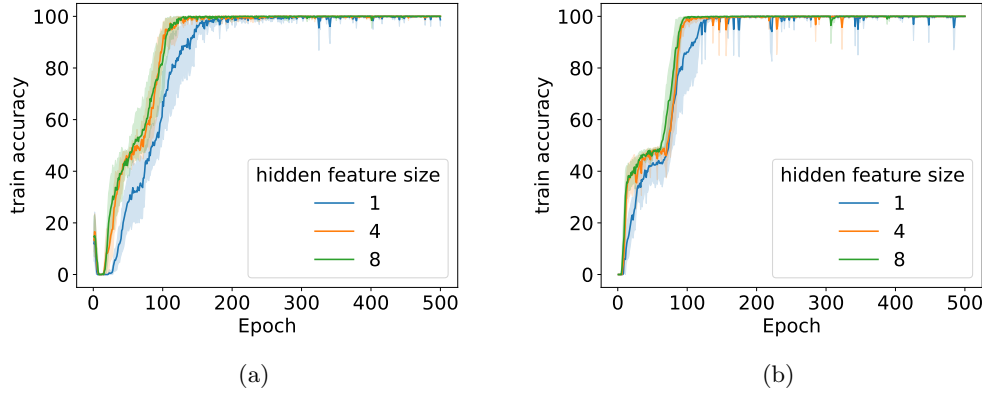


Figure 3.5: Experimental Framework E1. Training accuracy over the epochs for a DGNN trained on the dataset containing dynamic graphs up to time length $T = 4$ (a) and $T = 5$ (b).

Results. The results of the experiments confirm our theoretical statements. More precisely, the DGNNs performed as follows during training.

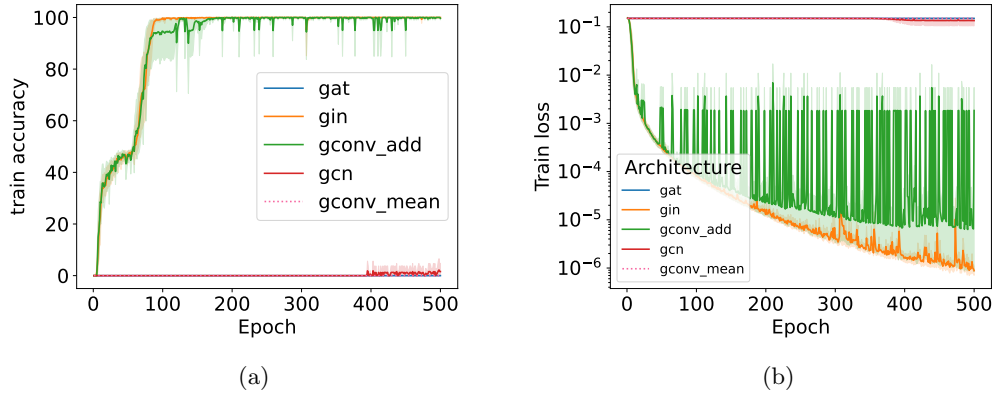


Figure 3.6: Experimental framework E2. Training accuracy a) and training loss b) over the epochs for several DGNNs trained on the dataset containing dynamic graphs up to time length $T = 5$. Figure b) is in logarithmic scale.

E1 In Fig. 3.5, the evolution of the training accuracy over the epochs is presented for different GIN hidden layer sizes h_{gin} and for dynamic graphs up to time lengths $T = 4$ (Fig. 3.5a) and $T = 5$ (Fig. 3.5b). All the architectures reach 100% accuracy for experiments on both time lengths. Even setting $h_{gin} = 1$ leads to a perfect classification at a slower rate. It may appear surprising that, even with a hidden representation of size 1, the DGNN can well approximate the function F_{DWL} .

However, as we already pointed out in paragraph 3.3.2, the possibility of reaching the universal approximation with a feature of dimension 1 is confirmed by The-

orem 3.8.

E2 The DGNN with the GIN module achieve the best performance in terms of learning accuracy and speed of decreasing, as illustrated in Fig. 3.6. The DGNN with the `gconv_add` module is able to learn the task, although learning is unstable (see Fig. 3.6 b)). This is not surprising since this module has been proven to match the expressive power of the 1-WL test [59]. The other DGNNs are incapable to learn the objective function. This is a consequence of their weaker expressive power, widely investigated in literature [26, 88].

Thus, overall, our theoretical expectations were met by both experiments.

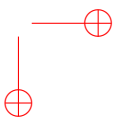
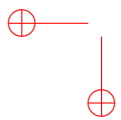
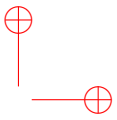
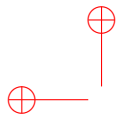
3.5 Conclusion

This chapter provided two extensions of the 1-WL isomorphism test and the unfolding tree. First, we introduced WL test notions to attributed and dynamic graphs, and second, we introduced extended concepts for unfolding trees on attributed and dynamic graphs. Further, we extended the strong connection between unfolding trees and the (dynamic/attributed) 1-WL tests, proving that they induce the same equivalence between nodes for the attributed and the dynamic case, respectively.

Regarding the models employed in the present investigation, AGNN and DGNN, we have proved that both the different GNN types can approximate in probability up to any precision, any function that preserves the (attributed/dynamic) unfolding equivalence.

Note that the dynamic GNN considered here follows a discrete-time representation, i.e., as a sequence of static graph snapshots without actual timestamps. Thus, Theorem 3.8 does not hold for all Dynamic GNNs, as we consider a discrete recurrent model working on graph snapshots (also known as Stacked DGNN). Nevertheless, several DGNNs of this kind are listed in [105], such as GCRN-M1 [108], RgCNN [109], PATCHY-SAN [110], DyGGNN [111], and others. Still, the approximation capability depends on the functions AGGREGATE and COMBINE designed for each GNN working on the single snapshot and the implemented Recurrent Neural Network. For example, the most general model, the original RNN, has been proven to be a universal approximator [106].

In the future, we could investigate other extensions like, for example, the n -dimensional attributed/dynamic WL test or other versions of unfolding trees, covering GNN models not considered by the frameworks used in this study. A preliminary work in this direction is presented in the next chapter.



Chapter 4

The Extended Weisfeiler-Leman Hierarchy for Arbitrary Graph Types

Due to the limitations of 1-WL, several more powerful extensions have been devised. One of the most natural is the k -WL, coloring k -tuples in \mathcal{V}^k instead of single nodes. The k -WL test forms a totally ordered chain where the power of the tests increases with the value of k . In particular, k -WL is less powerful than $(k+1)$ -WL for every $k > 1$ and 1-WL is equivalent to 2-WL. This result shows that if two graphs are not distinguishable by a k -WL test, there must be an l -WL test with $l > k$ that can distinguish them [61].

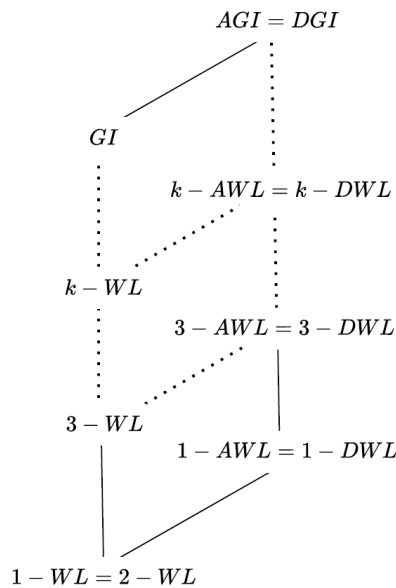


Figure 4.1: The WL complete lattice. Its minimum is the 1-WL test, equivalent to the 2-WL test, and its maximum is the Attributed Graph Isomorphism (AGI) and the Dynamic Graph Isomorphism (DGI) test.

Given that the k -WL tests only apply to simple node-attributed graphs, we first revise the attributed and dynamic extensions of the k -WL. Then we draw connections among the various versions of WL, creating a novel extended hierarchy of tests for arbitrary graphs (reported in Figure 4.1). The order among the tests is no longer total but *partial*, so we will employ basic concepts from lattice theory to analyze the resulting structure. A promising result is that the new WL-hierarchy constitutes a complete and distributive lattice with the partial order defined by: $A \leq B$ iff the partition \mathcal{P}_A induced by the test A is equal or coarser than the partition \mathcal{P}_B induced by the test B . The minimum element of this lattice is the 1-WL test, equivalent to the 2-WL test, while the maximum element is the Attributed Graph Isomorphism (AGI) test and Dynamic Graph Isomorphism (DGI) test. This chapter is based on [73] which is a preliminary work whose final goal is to find useful implications from lattice theory for the GI problem. Some possible research questions on this topic could be, e.g.,

- How big is the **difference** $|\mathcal{P}_B| - |\mathcal{P}_A|$ of the partitions $\mathcal{P}_A, \mathcal{P}_B$ if $A \leq B$?

- Which approaches from **lattice theory** can be applied to the WL hierarchy to infer beneficial consequences from it?
- Is it possible for two graphs to find the **minimal WL-test** capable of distinguishing the graphs?
- What are **minimal requirements** to a subset of WL-tests such that it remains a **lattice**, or that we obtain a **semilattice**?

4.1 Preliminaries from Lattice Theory

Here, we provide some preliminary notions from lattice theory such as the definition of lattice itself and some related properties. The mathematical structure of a lattice can be fully characterized using algebraic terms. The partial order coming from the WL-hierarchy extension, including the k -dimensional attributed and dynamic WL tests, results in a lattice of all the considered WL tests.

Definition 4.1 (Semilattice). Let $\mathcal{L} = \langle L, \cdot \rangle$ be a commutative semi-group. If $\forall a \in L$ holds $a \cdot a = a$ (i.e. every element in A is idempotent) then \mathcal{L} is a **semilattice**.

Definition 4.2 (Lattice). Let $\mathcal{L} = \langle L, \vee, \wedge \rangle$ be an algebra such that $\langle L, \vee \rangle$ and $\langle L, \wedge \rangle$ are semilattices.

\mathcal{L} is a **lattice** if the *absorption laws* hold, i.e. if

$$a \vee (a \wedge b) = a \quad \text{and} \quad a \wedge (a \vee b) = a \quad \forall a, b \in L.$$

The lattice operations \vee and \wedge are called *join* and *meet*, respectively. It is particularly important to underline the relation between lattices and partially ordered sets (posets). Let L be a set, \leq a partial order on L (i.e., \leq is reflexive, anti-symmetric and transitive) and $X \subseteq L$. Then, an element $a \in L$ is an upper bound of X in L , such that $x \leq a \forall x \in X$ and, if $x \leq b \forall x \in X$ then $a \leq b$, if it exists. Similarly, the lower bound can be defined. Given $X \subseteq L$, we will indicate with $\vee X$ and $\wedge X$ the upper bound and the lower bound respectively (if they exist).

Theorem 4.1. Let L be a poset such that any $X \subseteq L$, $|X| < \infty$, admits upper and lower bounds. Then, $\langle L, \vee, \wedge \rangle$ is a lattice where, $\forall a, b \in L$

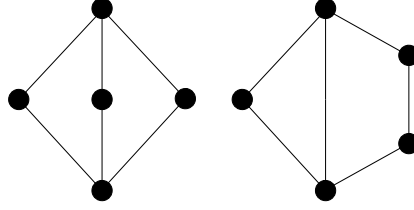
$$a \vee b = \bigvee \{a, b\} \quad \text{and} \quad a \wedge b = \bigwedge \{a, b\}.$$

Vice versa, if $\langle L, \vee, \wedge \rangle$ is a lattice, then the relation $a \leq b$ iff $a \vee b = b$ iff $a \wedge b = a$ is a partial order.

Every finite subset $X \subseteq L$, $X = \{a_1, \dots, a_n\}$ admits upper and lower bounds defined by

$$\bigvee X = a_1 \vee \dots \vee a_n \quad \bigwedge X = a_1 \wedge \dots \wedge a_n.$$

Definition 4.3. Let \mathcal{L} be a lattice. Then $a, b \in L$ are **comparable** if $a \leq b$ or $b \leq a$. Otherwise, they are **incomparable**.

Figure 4.2: Hasse diagram of the lattices M_3 (left) and N_5 (right).

If the elements of L are pairwise comparable, the lattice is fully ordered and it is called a **chain**. We say that b **covers** a if $\{c : a \leq c \leq b\} = \{a, b\}$, denoted as $a \preceq b$. The best way to understand the order structure of a lattice is to draw its *Hasse diagram*; Informally, in a Hasse diagram the lattice elements are arranged on a plane so that if $a < b$ then b is above a . Moreover, a line is drawn from a to b , whenever $a \prec b$.

Definition 4.4 (Complete Lattice). A lattice \mathcal{L} is **complete** if each subset of L admits upper and lower bounds.

Definition 4.5 (Distributive Lattice). A lattice \mathcal{L} is **distributive** if $\forall a, b, c \in \mathcal{L}$

$$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c).$$

There are two ‘minimal’ counterexamples to distributivity, namely the non-distributive lattices M_3 and N_5 , depicted in Figure 4.2.

The following theorem characterizes distributive lattices in terms of ‘forbidden substructures’.

Theorem 4.2 ([112]). *Let \mathcal{L} be a lattice. Then \mathcal{L} is distributive iff, \mathcal{L} does not contain an unbounded sublattice which is isomorphic to M_3 or N_5 .*

4.2 The extended Weisfeiler-Leman Hierarchy

Here we extend the k -WL hierarchy to arbitrary graph types, by defining an Attributed and dynamic k -WL tests.

Definition 4.6 (Attributed k -WL Test). Given a k -tuple $s = (s_1, \dots, s_k) \in \mathcal{V}^k$, the j -th neighborhood of s is defined as in 2.6. It is an inductive procedure: in iteration $i = 0$, the algorithm labels each k -tuple with its *atomic type*, i.e., two k -tuples s and s' in \mathcal{V}^k get the same color if the map $s_i \mapsto s'_i$ induces an attributed isomorphism between the subgraphs induced from the nodes from s and s' , respectively. For iteration $i > 0$, we define the list of weights of the j -th neighborhood of s as:

$$\Omega_{s,j} = \left(\omega_{(x,y)} \mid (x,y) \in \mathcal{E} \text{ and } x,y \text{ are in the same tuple in } N_{s,j} \right).$$

Then we set

$$C_{s,j}^{(i)} = \text{Hash} \left(\left(\mathbf{c}_{s'}^{(i-1)} \mid s' \in N_j(s) \right) \right),$$

and

$$\mathbf{c}_s^{(i)} = \text{Hash} \left(\mathbf{c}_s^{(i-1)}, (\Omega_{s,1}, \dots, \Omega_{s,k}), (C_{s,1}^{(i)}, \dots, C_{s,k}^{(i)}) \right).$$

Analogously to the k -WL test, the k -AWL proceeds as the 1-WL test and two tuples s and s' with $\mathbf{c}_s^{(i-1)} = \mathbf{c}_{s'}^{(i-1)}$ get different colors in iteration i if there exists $j \leq k$ such that the number of j -neighbors of s and s' , respectively, colored with a certain color is different. Additionally, an increasing k implies a more powerful k -AWL test.

For the dynamic WL test an extended version of the attributed 1-WL test is used and defined in Def. 3.5 in r -dimensions. Note, that it is not equal to an attributed extension of the r -dimensional WL-test defined in [59] since here a special subset of r nodes is used. Based on the modified WL test, the corresponding dynamic WL equivalence is defined respectively.

Definition 4.7 (Dynamic k -WL Test). Given a k -tuple $s = (s_1, \dots, s_k) \in \mathcal{V}_t^k$, the j -th neighborhood of s at time t is obtained by replacing the j -th component of s by every node from \mathcal{V}_t :

$$N_{s,j}(t) = \{(s_1, \dots, s_{j-1}, r, s_{j+1}, \dots, s_k) \mid r \in \mathcal{V}_t\}$$

At iteration $i = 0$, the algorithm labels each k -tuple with its *atomic type* analogously to the k -AWL, but for any $t \in I$. For iteration $i > 0$, we define the list of weights of the j -th neighborhood of s at time t as:

$$\Omega_{s,j} = \left(\omega_{\{x,y\}} \mid (x,y) \in \mathcal{E} \text{ and } x,y \text{ are in the same tuple in } N_{s,j}(t) \right).$$

Then we set

$$C_{s,j}^{(i)}(t) = \text{Hash}_t \left((\mathbf{c}_{s'}^{(i-1)}(t) \mid s' \in N_{s,j}(t)) \right),$$

and

$$\mathbf{c}_s^{(i)}(t) = \text{Hash}_t \left(\mathbf{c}_s^{(i-1)}(t), (\Omega_{s,1}(t), \dots, \Omega_{s,k}(t)), (C_{s,1}^{(i)}(t), \dots, C_{s,k}^{(i)}(t)) \right).$$

Note that the Hash_t function can be different in each timestamp.

4.2.1 Establishing connections between the various tests

In this section, the attributed and dynamic WL tests are positioned into the Weisfeiler-Leman Hierarchy.

Theorem 4.3. *1-WL test* \sqsubset *1-AWL test*

Proof. First, we show that 1-WL test is a subset of the 1-AWL test and then we give a counter example for them being equal.

\sqsubset Let $g_1 := (\mathcal{V}_1, \mathcal{E}_1)$, $g_2 := (\mathcal{V}_2, \mathcal{E}_2)$ be two graphs with $g_1 \sim_{1\text{-WL}} g_2$. Introducing empty attribute functions $\alpha_i : \mathcal{V}_i \rightarrow \emptyset$, $\omega_i : \mathcal{E}_i \rightarrow \emptyset$ for $i = 1, 2$ leads to the the same initial coloring of the nodes and unattributed edges, and thus to the performance of the 1-WL test. Therefore, it follows $g_1 \sim_{1\text{-AWL}} g_2$.

\neq In Fig. 4.3 one can see that the 1-AWL test can distinguish the first and last graph (graphs a and c), while the 1-WL test fails.

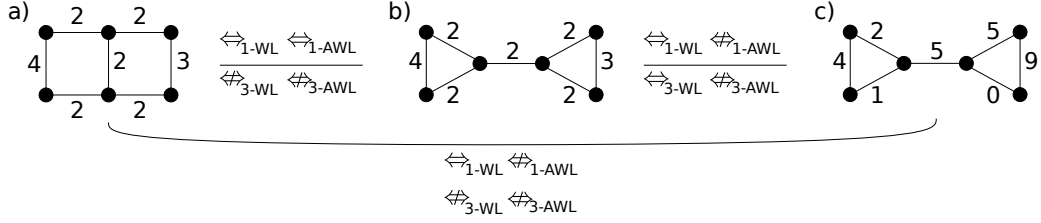


Figure 4.3: The graphs a) and b) are falsely recognized as 1-WL and 1-AWL isomorphic; the same holds for the 3-WL and 3-AWL. The graphs b) and c) are falsely recognized as isomorphic just by the 1-WL/3-WL test while the 1-AWL/3-AWL test correctly recognizes both graphs as non isomorphic. The same result holds for the graphs b) and c) for 1-WL and 1-AWL while 3-WL and 3-AWL can distinguish both. The 1-WL and 3-WL equivalences on the graphs from a) and b) and a) and c) without attributes are taken from [4, §3.1].

□

Corollary 4.4. *2-WL test \sqsubseteq 1-AWL test*

Proof. As stated in [113, §3.5, Cor. 3.5.8], 1-WL test and 2-WL test are equivalent. □

Corollary 4.5. *k-WL test \sqsubseteq k-AWL test.*

Proof. Follows analogously from the proof of Theorem 4.3. □

Corollary 4.6. *The k-DWL test \sqsubseteq (k+1)-DWL test.*

Proof. Follows immediately from the static Weisfeiler-Lemann Hierarchy. □

Theorem 4.7. *The 3-WL test and the 1-AWL test cannot be compared regarding to the Weisfeiler Lemann Hierarchy.*

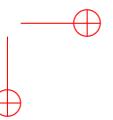
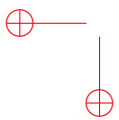
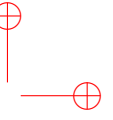
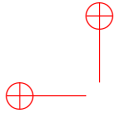
Proof. In Fig. 4.3 one can see that the 3-WL test can distinguish the graphs a) and b) but the 1-AWL test cannot. However, the 3-WL test cannot distinguish the graphs b) and c) while the 1-AWL test can. □

Theorem 4.8. *1-DWL \equiv 1-AWL*

The proof of the theorem can be found in the Appendix B.3.1.

Corollary 4.9 (Weisfeiler-Leman Lattice). *The Weisfeiler-Leman hierarchy is an infinite, bounded, complete, and distributive lattice (as a consequence of theorem 4.2). The partial order relation is: $A \sqsubseteq B$ iff the partition \mathcal{P}_A induced by the test A is equal or coarser than the partition \mathcal{P}_B induced by the test B, i.e. $|\mathcal{P}_A| \leq |\mathcal{P}_B|$ and each subset in \mathcal{P}_A is a union of subsets in \mathcal{P}_B , i.e.,*

$$\forall A \in \mathcal{P}_A \exists \{\mathcal{B}_i\} \subset \mathcal{P}_B, \text{ for some indices } i, \text{ s.t. } A = \bigcup \mathcal{B}_i.$$

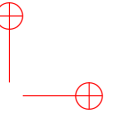
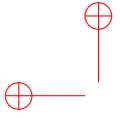


The minimum element of this lattice is the 1-WL test, equivalent to the 2-WL test, while the maximum element is the Attributed Graph Isomorphism test (AGI), equivalent to the Dynamic Graph Isomorphism test (DGI).

The lattice is represented in Figure 4.1.

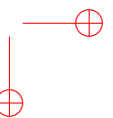
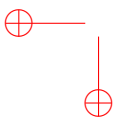
4.3 Conclusion

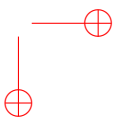
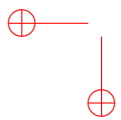
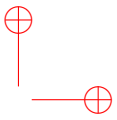
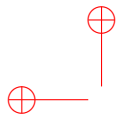
Graph isomorphism is still a non-trivial problem lying in the class NP. However, a common practical solution for at least distinguishing non-isomorphic graphs is the Weisfeiler-Lehman (WL) method. Given that the initial WL tests can just handle node-attributed graphs, in this preliminary work we extended the WL Hierarchy to arbitrary graphs. Particularly, we introduced a k -dimensional attributed and dynamic version of the k -WL test. We further investigated the relation between the devised tests and we found out that it is a partial ordering. This results in a complete and distributive lattice, which paves the way for concepts of lattice theory to be utilized in the context of graph isomorphism and WL tests.

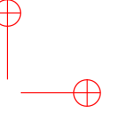
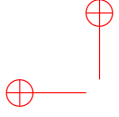


Part III

Going Beyond the Topological Neighborhood







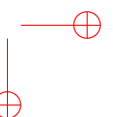
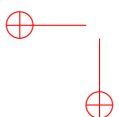
Chapter 5

Path-based Graph Neural Networks: A New Hierarchy of Highly Expressive GNNs

In this chapter, we systematically investigate the expressive power of *paths* to increase the expressivity of graph neural networks (GNNs). In particular, the message-passing graph neural network has been shown to be at most as expressive as the Weisfeiler-Leman (1-WL) color refinement algorithm [26, 59]. Given all the well-studied limitations of 1-WL, research efforts have resulted in several novel graph neural networks which leverage graph substructures to improve expressivity [114, 115, 116, 117, 118, 119, 120]. Paths are arguably one of the simplest graph substructures. Despite that, paths have only recently received attention in the context of GNNs, and can be broadly categorized into GNNs which incorporate shortest path information [121, 122, 123, 124, 125, 126, 127] and GNNs which aggregate or sample from the set of all paths [128, 129, 130]. While it has been shown that the incorporation of shortest path information increases the expressive power of GNNs [121, 122, 123], [130] demonstrated that shortest paths alone are not more expressive than 1-WL. Thus, it is of interest to investigate GNNs which consider all paths. Recently, [129] has proven that paths can be used to generalize topological neural networks such as CW Networks [131]. [128] sample from all paths and [130] aggregate over paths instead of the standard topological neighborhood. While it has been shown that path-based GNNs can achieve strong empirical performances in combination with distance encoding, a precise characterization of their expressive power is lacking. We propose to fill the existing gap in the literature and show that path-based GNNs with distance information form a novel class of highly expressive GNNs.

The content of the chapter is based on [74, 75] and outlined below.

- We propose PATH-WL, a general class of color refinement algorithms based on paths and geodesic distance information. PATH-WL is an iterative procedure that performs message passing on all paths up to a certain length.
- We prove that PATH-WL is strictly more expressive than 1-WL and we characterize graph classes that can be distinguished by PATH-WL.
- We demonstrate the ability of PATH-WL to count cycles of arbitrary length.
- We prove that PATH-WL is incomparable to the k -WL algorithm as well as to other expressive architectures.



- We eventually design PAIN, a GNN with expressive power equivalent to PATH-WL and empirically verify our theoretical results.

5.1 Preliminary definitions

Definition 5.1 (Path). A **path** $p = (v, v_1, \dots, v_\ell)$ in a graph $G = (\mathcal{V}, \mathcal{E})$, is a sequence of non-repeated nodes connected through edges in \mathcal{E} . The length of a path is the number of edges of the path or, equivalently, the number of nodes -1 . For example, the path $p = (v, v_1, \dots, v_\ell)$ has length ℓ .

We denote with $P_v^\ell := \{(v, v_1, \dots, v_k) \mid 1 \leq k \leq \ell\}$, the set of all the possible paths of length up to ℓ starting at node v . The edge case with $\ell = 0$ is $P_v^0 := \{v\}$.

Definition 5.2 (Cycle). A **cycle** $C_\ell = (v_1, \dots, v_\ell)$ of length ℓ is a sequence of adjacent and non-repeated nodes in the graph G , with the additional condition that the first and the last node are adjacent, i.e., there exists an edge (v_1, v_ℓ) . The length of a cycle is determined by the number of edges or the number of nodes in the cycle (which are the same).

In the following, some properties of graphs are formally defined. These properties are crucial to the understanding of the content of the chapter.

Definition 5.3 (Useful graph properties). Let $G = (\mathcal{V}, \mathcal{E})$ be a graph.

G is **connected** if for any $v, u \in \mathcal{V}$ there exists a path connecting u and v .

G is **d -regular** if every node has the same degree d , i.e., $\forall v \in \mathcal{V} \delta(v) = d$.

G is **strongly regular**, noted as $SR(n, k, \lambda, \mu)$, if G is a regular graph with n vertices and degree k such that (i) every two adjacent vertices have λ common neighbors, and (ii) every two non-adjacent vertices have μ common neighbors for some integers $\lambda, \mu \geq 0$.

G is **traceable** if it contains a Hamiltonian path, namely a path that includes all the nodes of the graph. A traceable graph is **homogeneously traceable** if every node of the graph is an endpoint of a Hamiltonian path.

G is **Hamiltonian** if it contains a Hamiltonian cycle, i.e., a cycle that contains every node in the graph exactly once. Hamiltonian graphs are homogeneously traceable, but the converse is not necessarily true.

The ability to count substructures plays a pivotal role in many applications. Thus, we formally define what we mean with *counting* and *recognizing* a pattern subgraph \mathcal{F} .

Given two graphs G and F , we write $sub(F, G)$ to denote the **number of subgraphs** of G isomorphic to F . Similarly, let $sub(F, G, u)$ be the number of subgraphs in G isomorphic to F which includes the node u .

Definition 5.4 (Counting). We say that a coloring algorithm T can **count** a substructure \mathcal{F} , if for two graphs G, H , $sub(\mathcal{F}, G) \neq sub(\mathcal{F}, H)$ implies that $T(G) \neq T(H)$.

Definition 5.5 (Recognizing). We say that a coloring algorithm T can **recognize** a substructure \mathcal{F} , if for two graphs G, H , $sub(\mathcal{F}, G) > 0$ and $sub(\mathcal{F}, H) = 0$ implies that $T(G) \neq T(H)$.

5.2 Path-WL: A Path-Based WL Test

In this section, we propose PATH-WL, a generalized class of color refinement algorithms to analyze the expressive power of path-based graph neural networks. The main difference between PATH-WL and 1-WL is that instead of aggregating over neighbors, PATH-WL aggregates over the multisets of paths. Furthermore, PATH-WL can use information about shortest path distances in the graph. In particular, for every node within a path of length ℓ , we combine the node color with the shortest path distance to the starting node. Note that we only add this information to every node in the path that is at most $d \leq \ell$ hops away from the starting node. Next, we define how to combine path information with shortest path information.

Definition 5.6. For a set of paths P_v^ℓ , we define the path multiset with distance encoding as

$$d\mathcal{P}_v^\ell := \left\{ \left\{ \left(\mathbf{c}_v, \eta_{vv}^d \right), \left(\mathbf{c}_{v_1}, \eta_{vv_1}^d \right), \dots, \left(\mathbf{c}_{v_k}, \eta_{vv_k}^d \right) \right\} \mid (v, v_1, \dots, v_k) \in P_v^\ell, k \leq \ell \right\},$$

where \mathbf{c}_v is the color of node v and $\eta_{vv_i}^d$ is the shortest path distance from v to v_i if the shortest path distance is less or equal to d and \emptyset otherwise.

Now we can introduce the iterative color refinement algorithm PATH-WL.

Definition 5.7 (PATH-WL). Let $G = (\mathcal{V}, \mathcal{E}, \alpha)$ be a graph with node coloring α . Let $\mathbf{c}_v^{(i)} \in \Sigma$ be the color of the node $v \in \mathcal{V}$ at iteration i , from a set of colors Σ . The initial color of the node v corresponds to the node coloring, that is $\mathbf{c}_v^{(0)} := \alpha_v$. Let **Hash** be an injective function encoding every path in $d\mathcal{P}_v^\ell$ with a color $\mathbf{c}_v \in \Sigma$, with $d \geq 0$. Then, the updating procedure for d -PATH-WL $^\ell$ is defined as

$$\mathbf{c}_v^{(i)} = \text{Hash} \left(d\mathcal{P}_v^{\ell, (i-1)} \right).$$

If needed, we make the number of iterations i explicit by writing d -PATH-WL $^{\ell, (i)}$. By increasing the shortest path length d in the distance encoding we can obtain increasingly expressive algorithms. Here, we highlight two variants of d -PATH-WL with interesting theoretical properties:

0-Path-WL. The simplest variant of PATH-WL is 0-PATH-WL which incorporates no additional distance information. Despite its simplicity, 0-PATH-WL is strictly more expressive than 1-WL, and a single iteration can distinguish a variety of graph families indistinguishable by 1-WL (see Section 5.3.3). As an example, consider the graphs G and G' in Figure 5.2. With uniform vertex features, these two graphs are indistinguishable by 1-WL. However, the first iteration of 0-PATH-WL can distinguish these graphs, as shown in the following proposition.

Proposition 5.1. For every $\ell \geq 5$, $G \approx_{0\text{-PATH-WL}^{\ell, (1)}} G'$, while $G \sim_{1\text{-WL}} G'$.

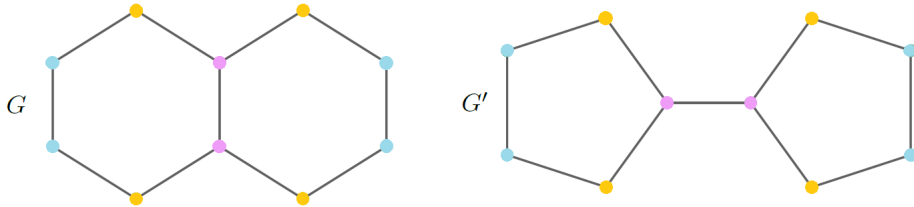


Figure 5.1: Stable graph coloring after one iteration of 1-WL. These two not isomorphic graphs represent the two real molecules Decalin and Bicyclopentyl [4].

Proof. Let $G = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ and $G' = (\mathcal{V}', \mathcal{E}', \mathbf{X}')$ be the graphs in Figure 5.2. The partition induced on nodes by 1-WL is represented in Figure 5.1, where each color corresponds to an equivalence class. We can see that there exists a bijection from the equivalence classes in G to the equivalence classes in G' and we can thus conclude $G \sim_{1\text{-WL}} G'$. For the other direction, first, let us notice the symmetry of the two graphs, where we can group the nodes into three different equivalence classes: (i) Nodes with degree 3, (ii) neighbors of degree 3 nodes, and (iii) nodes which are not adjacent to nodes with degree 3. We now consider the path multisets up to length 5, denoted by $0\text{-}\mathcal{P}^5$, for each of the three node equivalence classes. In Figure 5.2, we can see that $0\text{-}\mathcal{P}_v^5$ for nodes $v \in G, v' \in G'$ belonging to equivalence class (i) differ, as $0\text{-}\mathcal{P}_v^5$ contains 4 paths of length 5, while $0\text{-}\mathcal{P}_{v'}^5$ contains only 2 paths of length 5. Given the injectivity of the HASH function, for two nodes to get different colors it is sufficient to have different path multisets, as this ensures different colors from the first iteration onwards. The same argument applies to the remaining nodes in (ii) and (iii), which differ with respect to their multiplicity of paths of length 5: (ii) in G , these nodes have 5 paths of length 5, whereas in G' they have 3 paths of length 5 and for (iii) the nodes in G have 3 paths of length 5, whereas in G' they have 2 paths of length 5. Due to the different multiplicities of paths of length 5, we conclude that 0-PATH-WL is able to distinguish the two graphs. \square

Remark 5.1. Note that for path length equal to one it holds that $0\text{-PATH-WL}^1 \equiv 1\text{-WL}$.

1-Path-WL. We refer to the case of $d = 1$ as *neighbor marking*. This minor modification allows 1-PATH-WL to count cycles at the node level and suffices to prove that PATH-WL is not contained within the k -WL hierarchy. Furthermore, this implies that 0-PATH-WL is not bounded in expressivity by several other powerful GNNs such as sub-graph GNNs ([132, 119, 133]) or Local 2-GNNs ([134, 135, 136, 137]). See Section 5.3.1 and Section 5.3.2 for more details.

The expressive power of $d\text{-PATH-WL}^\ell$ is monotonically non-decreasing with respect to the path length ℓ and shortest path distance $d \leq \ell$.

Proposition 5.2. For every $d' \geq d \geq 0$ and $\ell' \geq \ell \geq 1$, it holds that

$$d\text{-PATH-WL}^\ell \subseteq d'\text{-PATH-WL}^\ell, \quad \text{and} \quad d\text{-PATH-WL}^\ell \subseteq d\text{-PATH-WL}^{\ell'}.$$

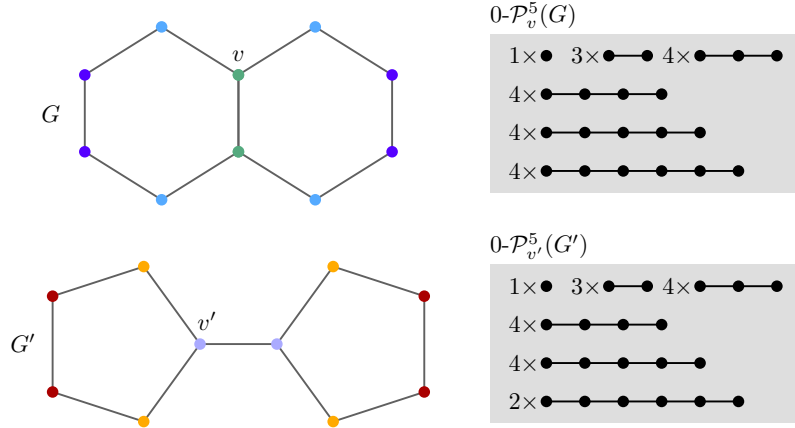


Figure 5.2: Example of non-isomorphic graphs for which 1-WL fails, but PATH-WL can distinguish them in the first iteration. In the grey boxes, we visualize the path multisets up to length 5 for nodes v and v' .

Refer to the Appendix C.3.1 for the proof of Proposition 5.2. Note that to ensure a monotonic increase of the expressive power with respect to the path length ℓ , it is important to consider the multiset of *all* paths *up to* length ℓ . Indeed, shorter paths can be crucial to distinguish two graphs, while longer paths can be identical.

We further point out that the least expressive variant 0-PATH-WL is at least as expressive as pathNN, the path-based GNN proposed by [130].

Remark 5.2. For every $\ell \geq 1$, $0\text{-PATH-WL}^{\ell,(\ell)} \supseteq \text{pathNN}$ ([130]) with path length ℓ .

Similarly to our architecture, pathNN pre-computes all paths of length up to ℓ . Then, it computes an embedding for each node and iteratively updates these embeddings by aggregating the embeddings of nodes in paths. Contrarily to our architecture, pathNN does not aggregate all path lengths at the same time and instead first aggregates over paths of length one, then over paths of length two, and so on. This limits the number of iterations to at most ℓ whereas 0-PATH-WL is not limited in the number of iterations. This is a strength of PATH-WL as we notice that increasing the number of iterations can reduce the path length needed for maximal expressivity (see Section 5.4). This can lead to a significant decrease in runtime as the runtime only scales linearly with the number of iterations but exponentially with the path length. We refer to the Appendix C.1 for a more in-depth discussion on Remark 5.2.

Time complexity. The time complexity of enumerating all possible paths of length at most ℓ for some fixed ℓ for one node in a graph G can be computed in $\mathcal{O}(\Delta^\ell)$ using depth-first-search, where Δ denotes the maximum vertex degree in G . For a graph of order n , this yields an overall time complexity of $\mathcal{O}(n\Delta^\ell)$ to compute all simple paths up to length ℓ . Note that we can perform the shortest path encoding with minimal computational

overhead, as shortest paths are a subset of all paths. Thus, i iterations of PATH-WL with path length ℓ has time complexity $\mathcal{O}(n\Delta^\ell i)$.

In practice, it is often not necessary to compute *all* paths. Instead, a small path length often suffices for maximal expressivity. Furthermore, real-world graphs are often sparse which drastically reduces the number of paths. In particular, real-world molecular datasets such as NCI1, NCI109, AIDS or PROTEINS ([138]) have an average degree of less than three which implies that only a small number of paths exists. Additionally, increasing the number of iterations can significantly decrease the path length required for maximal expressivity (see Section 5.4 and the Appendix C.7).

5.3 The Discriminative Power of Paths

In this section, we present our main results on the expressive power of path-based graph neural networks. For this, we analyze path-based graph neural networks within the mathematical framework of d -PATH-WL. First, in Section 5.3.1 we analyze how d -PATH-WL relates to k -WL. Then, in Section 5.3.2 we analyze its ability to count cycles. Finally, in Section 5.3.3 we characterize which graph families can be distinguished by d -PATH-WL for every $d \geq 0$ in only a single iteration. We provide a summary of our key theoretical results in Table 5.1.

Table 5.1: Summary of our theoretical results.

Theoretical Result	Requirement	Reference
$0\text{-PATH-WL}^\ell \equiv 1\text{-WL}$	$\ell = 1$	Rem. 5.1
$0\text{-PATH-WL}^{\ell,(\ell)} \supseteq \text{pathNN}^{(\ell)}$	$\forall \ell \geq 1$	Rem. 5.2
$0\text{-PATH-WL}^\ell \sqsupset 1\text{-WL}$	$\forall \ell > 1$	Thm. 5.3
$0\text{-PATH-WL}^{\ell,(1)}$ incomparable to $1\text{-WL}^{(\ell)}$	$\forall \ell \geq 3$	Thm. 5.11
$1\text{-PATH-WL}^\ell \not\sqsupseteq k\text{-WL}$	$\ell \sim k^2, k \geq 3$	Thm. 5.4 (1)
$k\text{-WL} \not\sqsupseteq 1\text{-PATH-WL}^\ell$	$\ell \geq 1, k \geq 3$	Thm. 5.4 (2)
1-PATH-WL^ℓ can count k -cycles	$\ell \geq k - 1$	Thm. 5.6
$1\text{-PATH-WL}^\ell \not\sqsupseteq \text{SubgraphGNN}$ $1\text{-PATH-WL}^\ell \not\sqsupseteq \text{Local 2-GNN}$ $1\text{-PATH-WL}^\ell \not\sqsupseteq \text{Folklore 2-GNN}$	$\ell \geq 7$	Thm. 5.8
$1\text{-PATH-WL}^\ell \not\sqsupseteq \text{Folklore } k\text{-GNN}$	$\ell \sim (k + 1)^2$	Thm. 5.8
$d\text{-PATH-WL}^\ell \sqsubseteq d'\text{-PATH-WL}^\ell$	$\ell \geq 1, d' \geq d \geq 0$	Prop. 5.2
$d\text{-PATH-WL}^\ell \sqsubseteq d\text{-PATH-WL}^{\ell'}$	$\ell' \geq \ell \geq 1, d \geq 0$	Prop. 5.2

5.3.1 Relation to the k -WL Hierarchy

Our first theorem states that for every path length and any shortest path distance, d -PATH-WL is more expressive than 1-WL.

Theorem 5.3. *For every path length $\ell > 1$, every $d \geq 0$, d -PATH-WL $^\ell$ is more expressive than 1-WL.*

Proof sketch. The proof consists of two parts: first, we prove that 0-PATH-WL is always at least as expressive as 1-WL. Then, we provide an example of two graphs G, G' and show the existence of two nodes $u \in \mathcal{V}_G$ and $v \in \mathcal{V}_{G'}$ that 0-PATH-WL can distinguish, but such that $u \sim_{1\text{-WL}} v$. The conclusion of the proof follows from the monotonicity of the expressive power of d -PATH-WL. Refer to C.4.1 in the Appendix for the full proof. \square

Theorem 5.3 states that even 0-PATH-WL, which simply aggregates over path multisets in each iteration, is more expressive than the standard Weisfeiler-Leman test. This implies that any GNN with expressive power greater than or equal to 0-PATH-WL is more expressive than the entire class of message-passing graph neural networks, since they are limited by 1-WL. We show an even stronger result on the relation to the k -WL hierarchy, for d -PATH-WL with $d \geq 1$:

Theorem 5.4. *Let $d \geq 1$ and $k \geq 3$. Then, d -PATH-WL and k -WL are incomparable. Equivalently, the following holds:*

- (1) *for every $k \geq 1$ there exists a path length ℓ such that d -PATH-WL $^\ell \not\sqsubseteq k$ -WL;*
- (2) *for every $\ell \geq 1$, there exists a k such that k -WL $\not\sqsubseteq d$ -PATH-WL $^\ell$.*

Proof sketch. The proof consists in finding for all k , a pair of non isomorphic graphs identified by k -WL but distinguished by d -PATH-WL with a sufficient path length, and viceversa. The first direction mainly relies on the fact that d -PATH-WL can distinguish between graphs with different cycle counts for any cycle length l (see Corollary 5.7) whereas k -WL fails to do so for cycles of length $l \sim k^2$ [139, Theorem 1.3]. We further show that one iteration and $d = 1$ is always sufficient to distinguish them.

For the other direction, fixed a certain length ℓ , we can always construct two graphs of treewidth 2, that can't be distinguished by d -PATH-WL $^\ell$ but can be distinguished by 3-WL [140, Theorem 6.1]. We refer to the Appendix for more details (cf. App. C.4.2). \square

Theorem 5.4 states that for every k , there exist graphs that d -PATH-WL can distinguish, while k -WL fails. This shows that d -PATH-WL is not limited by the k -WL hierarchy.

5.3.2 Counting Cycles

The expressive power of a test can also be described in terms of its ability to count substructures in the graph. Similar to [63, 141] we define the counting power of a test by its ability to distinguish between graphs with different substructure counts. We first show that 0-PATH-WL can distinguish between cycles of different lengths at node level.

Proposition 5.5. *Let C_n and C_m be two cycles of different lengths, with $n > m$. For any $v \in C_n$ and any $u \in C_m$,*

$$v \sim_{1\text{-WL}} u \quad \text{but} \quad v \not\sim_{0\text{-PATH-WL}^{\ell,(1)}} u \quad \forall \ell \geq m.$$

Proof. The left-hand side, i.e., $v \sim_{1\text{-WL}} u$, comes from the fact that every cycle is a connected 2-regular graph (cf. Appendix C.2) and that 1-WL cannot distinguish regular graphs at the node level. Let u and v be two arbitrary nodes in cycles C_n and C_m , respectively, and compute the first iteration of 0-PATH-WL for u and v . This consists of computing the multisets of paths \mathcal{P}_u and \mathcal{P}_v . Note that the longest path from each node in C_m has length $m - 1$ as C_m is a cycle. Since $n > m$, the multiset of paths from a node in C_n contains paths of length m so the two multisets \mathcal{P}_v^ℓ and \mathcal{P}_u^ℓ are different for every $\ell \geq m$. \square

Proposition 5.5 provides a theoretical justification for 0-PATH-WL to distinguish the graph instances in Figure 5.2, as they contain cycles of different lengths. In particular, the minimum path length necessary to discriminate the graphs in Figure 5.2 is exactly the size of the minimum cycle.

Our next theorem states that with the incorporation of neighborhood information, i.e., d -PATH-WL with $d \geq 1$, we can count cycles of arbitrary length at node level:

Theorem 5.6. *Let $\text{sub}(C_\ell, G, v) \neq \text{sub}(C_\ell, H, u)$ for some graphs G, H , nodes u, v and cycle C_ℓ . Then,*

$$u \approx_{1\text{-PATH-WL}^{\ell-1}} v.$$

Proof sketch. By Definition 5.6, neighbors v_i of the starting node v are identified by pairs $(\mathbf{c}_{v_i}, 1)$. Triangles on vertices v, v_1, v_2 are thus represented by paths of length 2 with two marked neighbors: $((\mathbf{c}_v, 0), (\mathbf{c}_{v_1}, 1), (\mathbf{c}_{v_2}, 1))$. These can be counted with $1\text{-PATH-WL}^{2,(1)}$. The general proof can be found in the Appendix C.5.1. \square

Note that being able to count cycles at node level is stronger than counting cycles at graph level. Indeed, node level counting implies graph level counting, but the opposite is not true. As a corollary of Theorem 5.6, 1-PATH-WL can distinguish between two graphs with different cycle counts:

Corollary 5.7. *Let $\text{sub}(C_\ell, G) \neq \text{sub}(C_\ell, H)$ for some graphs G, H and cycle C_ℓ . Then, $G \approx_{1\text{-PATH-WL}^{\ell-1}} H$.*

We can combine Corollary 5.7 with the analysis of [142] that investigates whether different GNNs can count cycles. This allows us to prove that PATH-WL is not bounded in expressivity by Subgraph GNNs ([132, 119, 133]), Local 2-GNNs ([134, 135, 136, 137]), and Folklore k -GNNs ([32, 136, 143]).

Corollary 5.8. *For every $\text{expGNN} \in \{\text{SubgraphGNN}, \text{Local 2-GNN}, \text{Folklore } k\text{-GNN}\}$, and every $k \geq 1$, there exists a path length ℓ such that $1\text{-PATH-WL}^\ell \not\sqsubseteq \text{expGNN}$.*

Please find more details in C.5.2 in the Appendix. While Corollary 5.7 is of great interest from a theoretical point of view ([62, 63, 141]), this result is also of practical relevance as it can reduce the path length needed to distinguish two graphs. For instance, we can distinguish the smallest pair of non-isomorphic strongly regular graphs *Rook* and *Shrikhande*, cf. Figure 5.3, with multisets of paths up to length 7, but paths of length up to 4 are sufficient if we use neighbor information, thus almost halving the required path length. This reduces the runtime by a factor of 200.

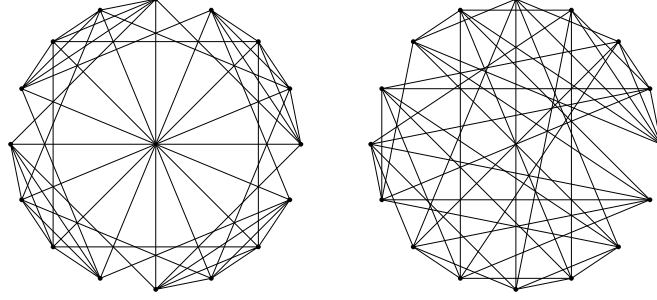


Figure 5.3: *Rook's 4x4 graph (left)* and the *Shrikhande (right)* graph. They cannot be distinguished by 3-WL but by 0-PATH-WL^{7,(1)} and 1-PATH-WL^{4,(1)}.

5.3.3 One Iteration Is Almost All You Need

In this section, we investigate graph families that can be distinguished by 0-PATH-WL with only *one* iteration. Formally, let \mathcal{G} and \mathcal{H} be two disjoint graph families. We say that a coloring algorithm T can distinguish \mathcal{G} and \mathcal{H} , if for every pair of graphs $G \in \mathcal{G}$ and $H \in \mathcal{H}$ it holds that $G \not\sim_T H$. Note that all results in this section generalize to d -PATH-WL as well as to pathNN ([130]).

Theorem 5.9. *There exists an ℓ such that 0-PATH-WL ^{$\ell,(1)$} can distinguish the following pairs of infinite graph families:*

1. *Hamiltonian graphs of different orders at node and graph level,*
2. *Hamiltonian graphs and non-homogeneously traceable graphs at graph level, and*
3. *almost all connected d -regular graphs and disconnected graphs with d -regular connected components at graph level.*

Proof sketch. For the proof, it is sufficient to show that path multisets can distinguish the graph families, as this is equivalent to 0-PATH-WL with one iteration. Please refer to Section C.6 in the Appendix for the full proof. \square

In Theorem 5.9, the families in (2) and (3) are of special interest, as they contain graph classes that are indistinguishable by 1-WL. For (2), consider the construction in Corollary 5.10. For (3), it is well known that 1-WL is not able to distinguish d -regular graphs. Examples for $d = 2$ and $d = 3$ can be found in Figure 5.4. Structures similar to Figure 5.4c could represent social networks, split into multiple communities. This example offers insights into the importance of distinguishing the two graphs, particularly in applications like community detection.

Corollary 5.10. *Let C_n be a cycle of length n . H is a graph of order $2n$ composed of two cycles C_{n+1} with an edge in common. G is a graph of order $2n$ composed of two cycles C_n connected by an extra edge. For any $n \geq 3$, it holds that*

$$G \sim_{0\text{-PATH-WL}^{2n-1,(1)}} H \quad \text{and} \quad G \not\sim_{1\text{-WL}} H.$$

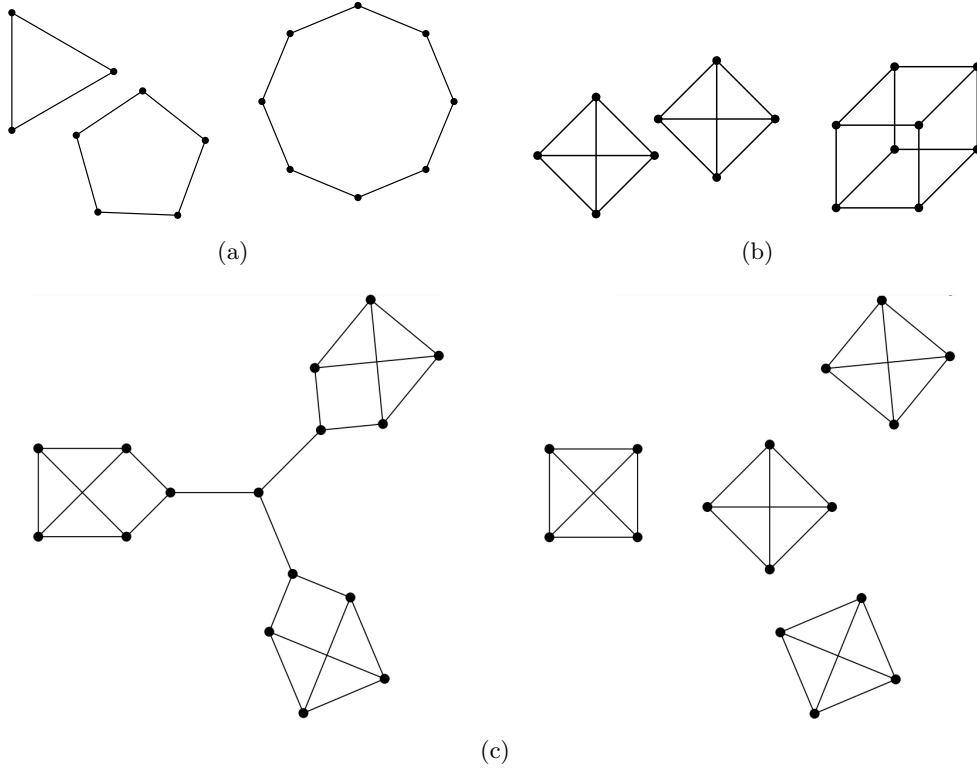


Figure 5.4: Graphs that cannot be distinguished by 1-WL but can be distinguished by 0-PATH-WL $^\ell$ with (a) $\ell = 3$ and (b),(c) $\ell = 4$. The required length ℓ corresponds to the order of the smallest connected component.

See Figure 5.5 for a visualization of how H and G are constructed. Note that such pairs of graphs are indistinguishable by 1-WL; the graphs in Figure 5.1 represent an instance of Corollary 5.10.

We have shown that even the simplest version of PATH-WL, 0-PATH-WL with one iteration, can already distinguish between graph families which are 1-WL-indistinguishable. However, the following theorem states that 0-PATH-WL with one iteration is not more expressive than 1-WL.

Theorem 5.11. *1-WL-equivalence at iteration ℓ and 0-PATH-WL $^{\ell,(1)}$ -equivalence are incomparable for every $\ell \geq 3$.*

Proof sketch. For the proof, it suffices to show that there exists a pair of nodes such that (i) 0-PATH-WL $^{\ell,(1)}$ is able to discriminate them, while 1-WL with ℓ iterations fails and (ii) vice-versa. For (i), please refer to Corollary 5.10 for an example. For (ii), please see the counterexample in Figure 5.6. See the Appendix C.6.2 for the complete proof. \square

Note that Figure 5.6 is a counterexample for a recent result of [130, Theorem 3.3]. 0-PATH-WL $^{3,(2)}$ is able to distinguish the graphs presented in Figure 5.6.

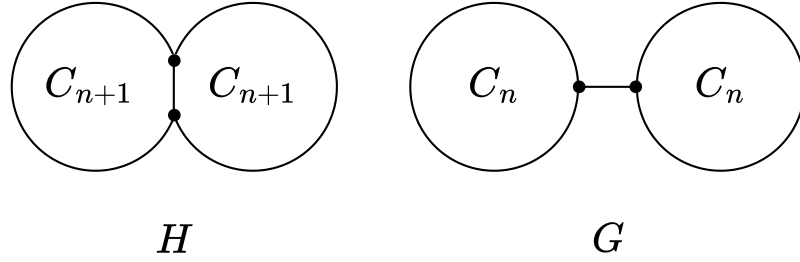


Figure 5.5: Two non-isomorphic 1-WL-equivalent graphs that can be distinguished by $0\text{-PATH-WL}^{2n-1,(1)}$.

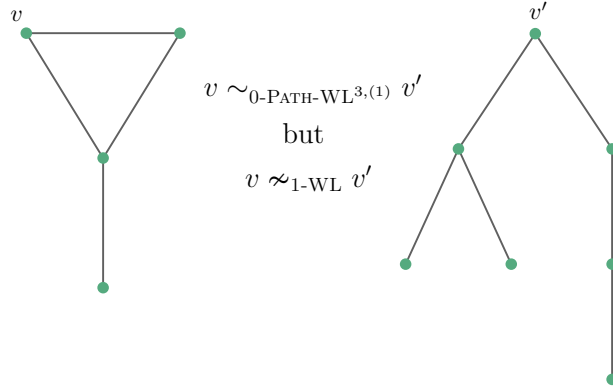


Figure 5.6: Counterexample that shows that path multisets alone are not more expressive than 1-WL.

5.4 Experimental Analysis

To empirically evaluate our findings, we design PAIN (PATH ISOMORPHISM NETWORK), a GNN architecture that is as powerful as PATH-WL. We evaluate PAIN on three datasets designed for studying the expressivity of GNNs and on one real-world benchmark dataset. We next define the PAIN family of GNNs.

5.4.1 The PAIN Family

Let $G = (\mathcal{V}, \mathcal{E}, \alpha)$ be a graph with node features α . Each GNN in the PAIN family has $n \geq 1$ layers, uses paths of length up to $\ell \geq 1$, and distances up to $d \leq \ell$. Analogously to PATH-WL, for a fixed distance d , we denote such a GNN as $d\text{-PAIN}$. PAIN computes an embedding $\mathbf{h}_v^{(i)}$ for each node $v \in \mathcal{V}$ in each layer $i \in \{1, \dots, n\}$. Similar to PATH-WL, we initialize each node embedding as the node features $\mathbf{h}_v^{(0)} = \alpha_v$. The embeddings are updated iteratively

$$\mathbf{h}_v^{(i)} = \text{AGGREGATE} \left(\left\{ \left\{ \mathbf{z}_p^{(i-1)} : p \in P_v^\ell \right\} \right\} \right)$$

where $\mathbf{z}_p^{(i-1)}$ is the embedding of path p defined via

$$\mathbf{z}_p^{(i-1)} = f(p) = f\left(\left(\mathbf{h}_v^{(i-1)}, \dots, \mathbf{h}_{v_\ell}^{(i-1)}\right)\right).$$

In order to gain maximal expressive power, the function f must be injective over sequences, and AGGREGATE must be injective over multisets. For f we select an LSTM ([144]) as they can approximate any function on sequences ([106]). In most experiments we use the sum for AGGREGATE as it allows the representation of injective functions over multisets [26, Lemma 5]. To get a graph-level prediction, we pool all node representations in the final layer and apply a multi-layer perceptron.

Datasets. To study the expressivity of the PAIN family, we use the synthetic datasets EXP ([145]), SR ([146]) and CSL ([147]). EXP contains 600 non-isomorphic pairs of graphs representing propositional formulas. Each pair of graphs in this dataset cannot be distinguished by 1-WL but can be distinguished by 3-WL. With SR we refer to the same subset of strongly regular graphs used by [130]. Each pair of graphs in this dataset cannot be distinguished by 3-WL, as 3-WL fails to distinguish strongly regular graphs. An instance of this dataset is visualized in Figure 5.3. CSL contains 150 graphs with 41 nodes belonging to 10 different isomorphism classes that are indistinguishable by 1-WL. These graphs are constructed by overlapping two different Hamiltonian cycles. Our experiments on EXP and SR evaluate to what degree untrained PAIN can distinguish graphs. For CSL, we train PAIN to predict the isomorphism classes. Additionally, we perform an ablation study to investigate the impact of iterations, path length and distance on the expressivity.

For real-world evaluation, we use ZINC ([148, 149]). ZINC contains 12,000 small molecules. For this dataset, PAIN performs a regression task to predict the solubility of each molecule.

Experimental setup. For EXP and SR, we closely follow the experimental setup of [130]. We use an untrained PAIN with a two-layer LSTM to compute 16-dimensional embeddings. We use Euclidean normalization on the input for the LSTM and consider two representations the same if the Euclidean distance is below $\epsilon = 10^{-5}$. Analogous to [130], for SR we restrict the path length to 4 due to computational considerations and use path length 5 for EXP. All presented results are repeated over 5 seeds. We use a one-layer 0-PAIN for EXP and a one-layer 1-PAIN for SR. For CSL, we perform stratified 5-fold cross-validation with a 3:1:1 split. We train a small PAIN model with an embedding dimension of 16. We train 500 epochs with a fixed learning rate of 10^{-5} . We report the test set accuracy in the epoch with the highest validation performance and average this test set accuracy over all cross-validation splits. We perform ablations for different values of the path length $\ell \in \{1, \dots, 6\}$, number of layers $n \in \{1, 2\}$, and distance encoding depth $d \in \{0, 1, \ell\}$.

On ZINC we train a 5 layer 1-PAIN with path length 3 and embedding dimension 128. As ZINC contains edge features, we extend PAIN accordingly. As common on ZINC, we train with an initial learning rate of 10^{-3} that we half whenever the validation metric does not increase for 20 epochs. The training stops after the learning rate dips below 10^{-5} or after

Table 5.2: Mean and standard deviation of accuracy (\uparrow) on the CSL dataset.

ℓ	1 Layer			2 Layers		
	0-PAIN	1-PAIN	2-PAIN	0-PAIN	1-PAIN	2-PAIN
2	12 ± 4	20 ± 0	20 ± 0	12 ± 4	64 ± 8	70 ± 9
3	18 ± 4	40 ± 0	50 ± 0	20 ± 0	47 ± 6	64 ± 4
4	29 ± 5	54 ± 5	90 ± 0	32 ± 3	64 ± 5	90 ± 0
5	50 ± 0	59 ± 1	100 ± 0	46 ± 5	67 ± 2	100 ± 0
6	50 ± 0	90 ± 0	100 ± 0	46 ± 5	90 ± 0	100 ± 0

Table 5.3: Pairs of graphs in EXP that cannot be distinguished by the given models. **Bold** marks the strongest result.

Model	EXP \downarrow
GIN ([26])	600
3-WL ([32])	0
pathNN ([130])	0
PAIN (ours)	0

1000 epochs. We train the model 10 times and report the average test set mean absolute error in the epoch with the lowest validation error. For more details on experiments see the Appendix C.8.

Distance Encoding. We do not use distance encoding for EXP. For SR we mark neighbors by adding a constant value of 1.0 to all the neighbors of the starting node within a path. For CSL and ZINC we use a different type of neighborhood encoding better suited for a learning task. For every distance encoding (even depth 0) we attach to each embedding in each path a learned vector that encodes some type of distance. In the case of $d \leq 1$, this vector encodes the position in the sequence. For the case of $d > 1$, this encodes the shortest path distance between the starting node and the current node in the path. Finally, when $d \geq 1$ we add an additional feature to the embedding of each node in the path that is 1 if it is a neighbor to the starting node in the path and 0 otherwise.

Our code can be found at <https://anonymous.4open.science/r/pathGNNs-D065/>.

Results. On EXP, untrained 0-PAIN with path length 5 can distinguish all graphs (Table 5.3), which is consistent with the results in [130]. Please note that we, however, do not use distance encoding as proposed in [130] and could thus verify that the multiplicities of paths of length 5 are sufficient. On SR, untrained 1-PAIN is able to distinguish more than 50% of all graph pairs (cf. Figure 5.7). In general, our results are comparable with [130] with distance encoding. For SR(29,14,6,7) we obtain significantly better results of around 40% failure rate in comparison to the 80% failure rate of pathNN.

Table 5.2 shows the results on the CSL dataset, which suggest that the expressivity increases with the number of layers, the path length, and the depth of the distance

Table 5.4: Mean and standard deviation of mean absolute error on the ZINC (12,000 nodes with edge features) dataset. GIN, GCN and GAT experiments were conducted by [5], all other GNNs were benchmarked by the cited authors. **Bold** marks the strongest architecture.

Model	MAE (\downarrow)
GIN ([26])	0.387 ± 0.015
GCN ([150])	0.278 ± 0.003
GAT ([24])	0.384 ± 0.007
CIN ([131])	0.079 ± 0.006
ESAN ([151])	0.102 ± 0.003
pathNN ([130])	0.090 ± 0.004
PAIN (ours)	0.148 ± 0.003

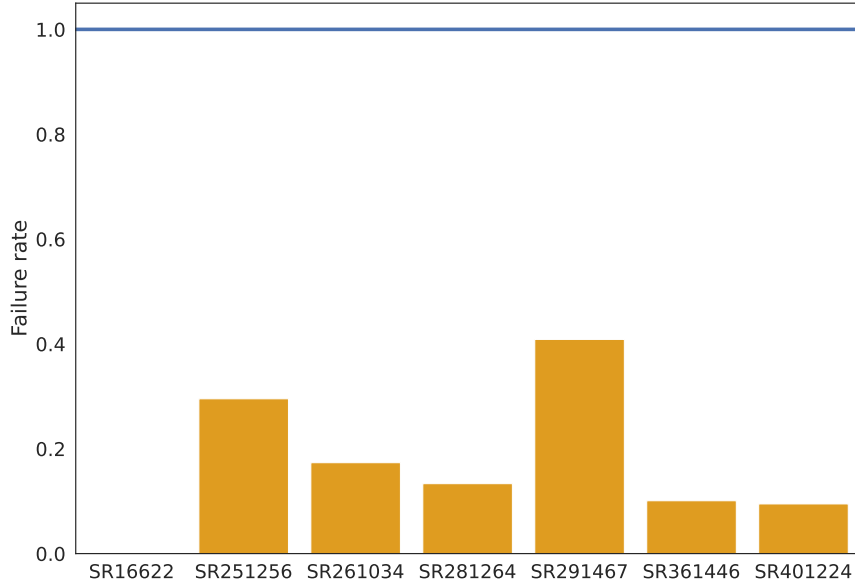
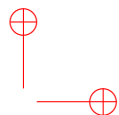
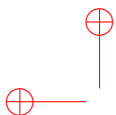


Figure 5.7: Results on SR for path length 4 and marked neighbors. The blue line indicates the failure rate of 3-WL.

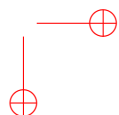
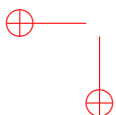
encoding. Most importantly, increasing the depth d gives a strong boost in accuracy without increasing the time complexity. For example for a single layer and path length $\ell = 5$, increasing the distance encoding depth from 0 to ℓ improves the accuracy from 50% to 100%. What is especially interesting is that no 0-PAIN model achieves an accuracy of over 50% for CSL which indicates that the distance encoding is crucial for the expressive power required for this dataset. Finally, we can see that increasing the number of layers gives a strong boost in predictive performance for short path lengths and especially for $\ell = 2$. This is important, as increasing the number of layers increases the runtime only by a constant factor compared with an exponential increase with ℓ . Finally, Table 5.4

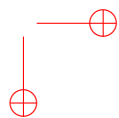
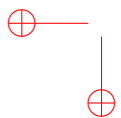
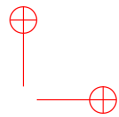
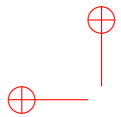


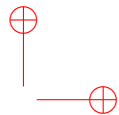
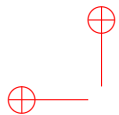
shows the results on ZINC. We observe that 1-PAIN outperforms classical message-passing GNNs.

5.5 Conclusion

In this chapter, we investigated the discriminative power of paths to distinguish between non-isomorphic graph instances. For this, we have proposed PATH-WL a general class of color refinement algorithms that allows us to investigate the expressivity of path-based graph neural networks. We have proven that PATH-WL is incomparable in expressivity to several other powerful GNNs such as subgraph GNNs or k -GNNs. Furthermore, PATH-WL is able to count cycles which is important for learning tasks on molecular structures. All of this indicates that path-based GNNs form a novel class of highly expressive GNNs. As the computational cost of our approach is strongly dependent on the chosen path length, we plan to characterize graph classes for which it is proven that a reasonably low path length is sufficient for maximal expressivity. Furthermore, we intend to investigate approaches that only require the computation of a subset of all paths of a certain length.

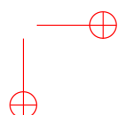
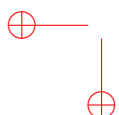


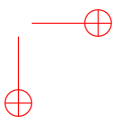
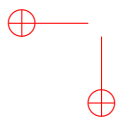
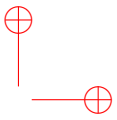
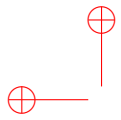




Part IV

Conclusions and Future Perspectives







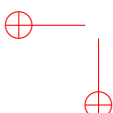
Chapter 6

General Conclusions

In this thesis, we aimed to extend the expressiveness of graph neural networks beyond the constraints enforced by the standard 1-WL test. Since the conclusions are already presented at the end of each chapter, we provide a summary and discuss future perspectives.

In the first part of the thesis, we addressed the lack of fundamental theory on data structures that do not belong to the class of static, undirected, and homogeneous graphs with node attributes. In particular, we suggested an alternative approach to process several graph domains with a unique and universal AGNN model. We introduced appropriate versions of 1-WL/UT equivalence, and, considering generic GNN models that can operate on those domains, we proved their universal approximation capabilities. Specifically, we proved that AGNNs and DGNNs can approximate in probability up to any precision, any measurable function that preserves the (attributed/dynamic) unfolding equivalence. Moreover, these results hold for graphs whose attributes are vectors of reals, and with completely unconstrained target functions. Further, we extended the higher-order attributed/dynamic k -WL hierarchy to arbitrary graphs and compared the various tests. The order among them is no longer total but *partial*, which prepares the ground for concepts of *Lattice Theory* to intersect with Graph Theory and Graph Neural Networks.

A special attention was also devoted to discrete dynamic GNNs, which are widely used in practice but lack a comprehensive foundational theory. Since the presented results only apply to discrete recurrent models working on graph snapshots, and not to all dynamic GNNs, future work will focus on extending the investigation to graphs with a continuous-time representation. One difficulty in this context lies in deciding in which sense two continuous-time dynamic graphs are called WL equivalent, since there are many possibilities for dealing with the given timestamps. For example, the investigation on the equivalence between dynamic graphs involves analyzing how to manage dynamic graphs that are equal in their structure but differ in their temporal occurrence, i.e., according to the commitment of the WL equivalence or the unfolding tree equivalence, it is required to decide whether the concepts need to be **time-invariant**. In the case when two graphs with the same structure must be distinguished if they appear at different times, the node and edge attributes can be extended by an additional dimension carrying the exact timestamp. Thereby, the unfolding trees of two (structurally) equal nodes would be different, having different timestamps in their attributes. Then, all dynamic graphs $G \in \mathcal{D}$ are defined over the same time interval I . This assumption can be made without loss of generality, since the set of timestamps of G can be padded by including missing timestamps \bar{t} and G can be padded by empty graphs $G_{\bar{t}} = \emptyset$.



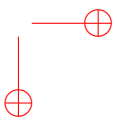
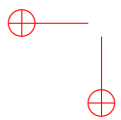
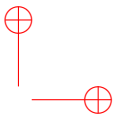
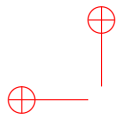
In the second part of the thesis, we searched for a new aggregation mechanism, to alleviate the short-sightedness of standard GNNs. To overcome the local neighborhood aggregation, we proposed using paths. In a broader sense, we systematically investigated the expressive power of path-based graph neural networks.

To this aim, we proposed PATH-WL, a new coloring algorithm based on paths and geodesic distances. We characterized families of graphs that can be distinguished by PATH-WL. For a sufficient path length, PATH-WL is not comparable to a wide range of expressive graph neural networks, it can count cycles, and achieves strong results on the notoriously difficult family of strongly regular graphs. Our theoretical results indicate that PATH-WL forms a new hierarchy of highly expressive graph neural networks. Even if the gained expressivity comes at an increased computational cost, we observed that the complexity is a function of the maximum degree of the graph, hence it is tractable for sparse graphs and it is strongly dependent on the chosen path length. We plan to characterize graph classes for which it is proven that a reasonably low path length is sufficient for maximal expressivity. We also aim to further explore how iterations (i.e. number of layers) can reduce the required path length, starting from the provided insights. Furthermore, we intend to investigate approaches that only require the computation of a subset of all paths, including various sampling strategies, to assess which theoretical guarantees are maintained in a probabilistic scenario.

A common future perspective unifying the two main lines of research in this thesis is to create a comprehensive hierarchy of WL tests, including PATH-WL and other more expressive algorithms, all adapted to deal with arbitrary graph types. A valuable addition would be to categorize the tests based on their ability to count substructures. This would transform the hierarchy into a practical guide for selecting the most suitable architecture for the problem at hand. Such extensions might result in a deeper understanding of the expressive power of different GNN architectures.

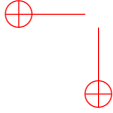
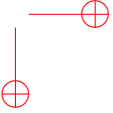
The present thesis focuses exclusively on the expressive power of GNNs. However, multiple factors are interleaved at the same time. GNNs with the same expressive power may differ for other fundamental properties, e.g., computational complexity and generalization capability. Indeed, more expressive models are typically more complex and can perfectly model the training data, including noise and outliers, which can lead to overfitting. This can make them less effective at predicting new, unseen data. Conversely, a less expressive model might underfit the training data, failing to capture essential patterns but potentially generalizing better, avoiding learning noise. Moreover, we discussed how increasing the number of layers can improve expressiveness (and in the case of path GNNs, decrease the required path length) but this can lead to over-smoothing. The issue of over-smoothing in GNNs is closely tied to the question of their expressiveness. This phenomenon occurs when multiple layers in a GNN repeatedly apply the same smoothing operation – typically, averaging node features with those of their neighbors. As a result, as the number of layers increases, the node features across the entire graph tend to converge to a similar state. This makes it difficult for the model to distinguish between nodes based on their features, effectively reducing the discriminative power of the network.

Therefore, as a future work, we also plan to understand how the architecture of AGGREGATE⁽ⁱ⁾, COMBINE⁽ⁱ⁾, and READOUT influence those other fundamental properties. Understanding and managing the interplay between expressiveness and generalization of GNNs is key factor to enhance their capabilities, making them more effective and versatile across a broad range of applications.





Bibliography

- [1] A.-L. Barabási, N. Gulbahce, and J. Loscalzo, “Network medicine: a network-based approach to human disease,” *Nature reviews genetics*, vol. 12, no. 1, pp. 56–68, 2011.
 - [2] A.-L. Barabasi and Z. N. Oltvai, “Network biology: understanding the cell’s functional organization,” *Nature reviews genetics*, vol. 5, no. 2, pp. 101–113, 2004.
 - [3] S. Beddar-Wiesing, G. A. D’Inverno, C. Graziani, V. Lachi, A. Moallem-Oureh, F. Scarselli, and J. M. Thomas, “Weisfeiler–lehman goes dynamic: An analysis of the expressive power of graph neural networks for attributed and dynamic graphs,” *Neural Networks*, p. 106213, 2024.
 - [4] R. Sato, “A survey on the expressive power of graph neural networks,” *arXiv preprint arXiv:2003.04078*, 2020.
 - [5] V. P. Dwivedi, C. K. Joshi, A. T. Luu, T. Laurent, Y. Bengio, and X. Bresson, “Benchmarking graph neural networks,” *Journal of Machine Learning Research*, vol. 24, no. 43, pp. 1–48, 2023.
 - [6] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, pp. 436–444, 2015.
 - [7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.
 - [8] M. O. Topal, A. Bas, and I. van Heerden, “Exploring transformers in natural language generation: Gpt, bert, and xlnet,” *arXiv preprint arXiv:2102.08036*, 2021.
 - [9] A. Gillioz, J. Casas, E. Mugellini, and O. Abou Khaled, “Overview of the transformer-based models for nlp tasks,” in *2020 15th Conference on Computer Science and Information Systems (FedCSIS)*. IEEE, 2020, pp. 179–183.
 - [10] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, “Geometric deep learning: going beyond euclidean data,” *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.
 - [11] D. Lazer, A. Pentland, L. Adamic, S. Aral, A.-L. Barabási, D. Brewer, N. Christakis, N. Contractor, J. Fowler, M. Gutmann *et al.*, “Computational social science,” *Science*, vol. 323, no. 5915, pp. 721–723, 2009.
 - [12] B. Yu, H. Yin, and Z. Zhu, “Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting,” *arXiv preprint arXiv:1709.04875*, 2017.
 - [13] M. Tubaishat and S. Madria, “Sensor networks: an overview,” *IEEE potentials*, vol. 22, no. 2, pp. 20–23, 2003.
- 
- 

- [14] E. H. Davidson, J. P. Rast, P. Oliveri, A. Ransick, C. Calestani, C.-H. Yuh, T. Minokawa, G. Amore, V. Hinman, C. Arenas-Mena *et al.*, “A genomic regulatory network for development,” *science*, vol. 295, no. 5560, pp. 1669–1678, 2002.
- [15] W. L. Hamilton, R. Ying, and J. Leskovec, “Representation learning on graphs: Methods and applications,” *arXiv preprint arXiv:1709.05584*, 2017.
- [16] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, “The graph neural network model,” *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.
- [17] C. Gallicchio and A. Micheli, “Graph echo state networks,” in *The 2010 international joint conference on neural networks (IJCNN)*. IEEE, 2010, pp. 1–8.
- [18] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, “Representation learning on graphs with jumping knowledge networks,” in *International conference on machine learning*. PMLR, 2018, pp. 5453–5462.
- [19] A. Micheli, “Neural network for graphs: A contextual constructive approach,” *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 498–511, 2009.
- [20] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, “Gated graph sequence neural networks,” *arXiv preprint arXiv:1511.05493*, 2015.
- [21] J. Bruna and S. Mallat, “Invariant scattering convolution networks,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1872–1886, 2013.
- [22] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” *arXiv preprint arXiv:1609.02907*, 2016.
- [23] W. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” *Advances in neural information processing systems*, vol. 30, 2017.
- [24] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph Attention Networks,” in *ICLR*, 2018.
- [25] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner *et al.*, “Relational inductive biases, deep learning, and graph networks,” *arXiv preprint arXiv:1806.01261*, 2018.
- [26] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, “How powerful are graph neural networks?” in *International Conference on Learning Representations*, 2019.
- [27] K. Hornik, “Approximation capabilities of multilayer feedforward networks,” *Neural networks*, vol. 4, no. 2, pp. 251–257, 1991.
- [28] G. Cybenko, “Approximation by superpositions of a sigmoidal function,” *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.
- [29] H. Maron, E. Fetaya, N. Segol, and Y. Lipman, “On the universality of invariant networks,” in *International conference on machine learning*. PMLR, 2019, pp. 4363–4371.
- [30] W. Azizian and M. Lelarge, “Expressive power of invariant and equivariant graph neural networks,” *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021. [Online]. Available: <https://openreview.net/forum?id=lxHgXYN4bwl>
- [31] N. Keriven and G. Peyré, “Universal invariant and equivariant graph neural networks,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [32] H. Maron, H. Ben-Hamu, H. Serviansky, and Y. Lipman, “Provably powerful graph networks,” *Advances in neural information processing systems*, vol. 32, 2019.

- [33] S. Jegelka, “Theory of graph neural networks: Representation and learning,” in *The International Congress of Mathematicians*, 2022.
- [34] J. Hartmanis, “Computers and intractability: a guide to the theory of np-completeness (michael r. garey and david s. johnson),” *Siam Review*, vol. 24, no. 1, p. 90, 1982.
- [35] H. A. Helfgott, J. Bajpai, and D. Dona, “Graph isomorphisms in quasi-polynomial time,” *arXiv preprint arXiv:1710.04574*, 2017.
- [36] R. C. Read and D. G. Corneil, “The graph isomorphism disease,” *Journal of graph theory*, vol. 1, no. 4, pp. 339–363, 1977.
- [37] H. R. Lewis, “Michael r. πgarey and david s. johnson. computers and intractability. a guide to the theory of np-completeness. wh freeman and company, san francisco1979, x+ 338 pp.” *The Journal of Symbolic Logic*, vol. 48, no. 2, pp. 498–500, 1983.
- [38] R. M. Karp, *Reducibility among combinatorial problems*. Springer, 2010.
- [39] S. Fortin, “The graph isomorphism problem,” 1996.
- [40] V. N. Zemlyachenko, N. M. Korneenko, and R. I. Tyshkevich, “Graph isomorphism problem,” *Journal of Soviet Mathematics*, vol. 29, pp. 1426–1481, 1985.
- [41] J. Kobler, U. Schöning, and J. Torán, *The graph isomorphism problem: its structural complexity*. Springer Science & Business Media, 2012.
- [42] M. Grohe and P. Schweitzer, “The graph isomorphism problem,” *Communications of the ACM*, vol. 63, no. 11, pp. 128–134, 2020.
- [43] L. Babai, “Graph isomorphism in quasipolynomial time,” in *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, 2016, pp. 684–697.
- [44] S. Kiefer, “Power and limits of the weisfeiler-leman algorithm,” Ph.D. dissertation, Dissertation, RWTH Aachen University, 2020, 2020.
- [45] D. G. Corneil and D. G. Kirkpatrick, “A theoretical analysis of various heuristics for the graph isomorphism problem,” *SIAM Journal on Computing*, vol. 9, no. 2, pp. 281–297, 1980.
- [46] S. Kiefer and D. Neuen, “A study of weisfeiler-leman colorings on planar graphs,” *arXiv preprint arXiv:2206.10557*, 2022.
- [47] S. Kiefer, I. Ponomarenko, and P. Schweitzer, “The weisfeiler-leman dimension of planar graphs is at most 3,” *Journal of the ACM (JACM)*, vol. 66, no. 6, pp. 1–31, 2019.
- [48] L. Babai, P. Erdos, and S. M. Selkow, “Random graph isomorphism,” *SIAM Journal on computing*, vol. 9, no. 3, pp. 628–635, 1980.
- [49] B. D. McKay *et al.*, “Practical graph isomorphism,” 1981.
- [50] B. Weisfeiler and A. Leman, “The reduction of a graph to canonical form and the algebra which appears therein,” *nti, Series*, vol. 2, no. 9, pp. 12–16, 1968.
- [51] H. L. Morgan, “The generation of a unique machine description for chemical structures—a technique developed at chemical abstracts service.” *Journal of chemical documentation*, vol. 5, no. 2, pp. 107–113, 1965.
- [52] G. W. Adamson and J. A. Bush, “A method for the automatic classification of chemical structures,” *Information Storage and Retrieval*, vol. 9, no. 10, pp. 561–568, 1973.
- [53] C. Morris, Y. Lipman, H. Maron, B. Rieck, N. M. Kriege, M. Grohe, M. Fey, and K. Borgwardt, “Weisfeiler and leman go machine learning: The story so far,” *The Journal of Machine Learning Research*, vol. 24, no. 1, pp. 15 865–15 923, 2023.

- [54] L. Babai and L. Kucera, “Canonical labelling of graphs in linear average time,” in *20th annual symposium on foundations of computer science (sfcs 1979)*. IEEE, 1979, pp. 39–46.
- [55] W. Li, H. Saidi, H. Sanchez, M. Schäfer, and P. Schweitzer, “Detecting similar programs via the weisfeiler-leman graph kernel,” in *Software Reuse: Bridging with Social-Awareness: 15th International Conference, ICSR 2016, Limassol, Cyprus, June 5-7, 2016, Proceedings 15*. Springer, 2016, pp. 315–330.
- [56] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, “Weisfeiler-lehman graph kernels.” *Journal of Machine Learning Research*, vol. 12, no. 9, 2011.
- [57] V. Arvind, J. Köbler, G. Rattan, and O. Verbitsky, “On the power of color refinement,” in *Fundamentals of Computation Theory: 20th International Symposium, FCT 2015, Gdańsk, Poland, August 17-19, 2015, Proceedings 20*. Springer, 2015, pp. 339–350.
- [58] S. Kiefer, “The weisfeiler-leman algorithm: an exploration of its power,” *ACM SIGLOG News*, vol. 7, no. 3, pp. 5–27, 2020.
- [59] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, “Weisfeiler and leman go neural: Higher-order graph neural networks,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 4602–4609.
- [60] C. Morris, F. Geerts, J. Tönshoff, and M. Grohe, “Wl meet vc,” in *International Conference on Machine Learning*. PMLR, 2023, pp. 25 275–25 302.
- [61] J.-Y. Cai, M. Fürer, and N. Immerman, “An optimal lower bound on the number of variables for graph identification,” *Combinatorica*, vol. 12, no. 4, pp. 389–410, 1992.
- [62] V. Arvind, F. Fuhlbrück, J. Köbler, and O. Verbitsky, “On weisfeiler-leman invariance: Subgraph counts and related graph properties,” *Journal of Computer and System Sciences*, vol. 113, pp. 42–59, 2020.
- [63] Z. Chen, L. Chen, S. Villar, and J. Bruna, “Can graph neural networks count substructures?” *Advances in neural information processing systems*, vol. 33, pp. 10 383–10 395, 2020.
- [64] L. H. Hall and L. B. Kier, “Molecular connectivity and substructure analysis,” *Journal of pharmaceutical sciences*, vol. 67, no. 12, pp. 1743–1747, 1978.
- [65] A. Prat-Pérez, D. Dominguez-Sal, J.-M. Brunat, and J.-L. Larriba-Pey, “Put three and three together: Triangle-driven community detection,” *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 10, no. 3, pp. 1–42, 2016.
- [66] H. Yin, A. R. Benson, J. Leskovec, and D. F. Gleich, “Local higher-order graph clustering,” in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 555–564.
- [67] C. E. Tsourakakis, J. Pachocki, and M. Mitzenmacher, “Scalable motif-aware graph clustering,” in *Proceedings of the 26th International Conference on World Wide Web*, 2017, pp. 1451–1460.
- [68] V. Satuluri, S. Parthasarathy, and Y. Ruan, “Local graph sparsification for scalable clustering,” in *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, 2011, pp. 721–732.
- [69] A. R. Benson, D. F. Gleich, and J. Leskovec, “Higher-order organization of complex networks,” *Science*, vol. 353, no. 6295, pp. 163–166, 2016.

- [70] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, "Network motifs: simple building blocks of complex networks," *Science*, vol. 298, no. 5594, pp. 824–827, 2002.
- [71] M. E. Newman, "The structure and function of complex networks," *SIAM review*, vol. 45, no. 2, pp. 167–256, 2003.
- [72] A. Krebs and O. Verbitsky, "Universal covers, color refinement, and two-variable counting logic: Lower bounds for the depth," in *2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science*. IEEE, 2015, pp. 689–700.
- [73] S. Beddar-Wiesing, G. A. D’Inverno, C. Graziani, V. Lachi, A. Moallem-Oureh, and franco scarselli, "On the extension of the weisfeiler-lehman hierarchy by WL tests for arbitrary graphs," in *18th International Workshop on Mining and Learning with Graphs*, 2022. [Online]. Available: <https://openreview.net/forum?id=Qt6GrgDz2y5>
- [74] C. Graziani, T. Drucks, F. Jogl, M. Bianchini, F. Scarselli, and T. Gärtner, "The expressive power of path-based graph neural networks," in *Forty-first International Conference on Machine Learning*, 2024. [Online]. Available: <https://openreview.net/forum?id=io1XSRtcO8>
- [75] C. Graziani, T. Drucks, M. Bianchini, F. Scarselli, T. Gärtner *et al.*, "No pain no gain: More expressive gnns with paths," in *NeurIPS 2023 Workshop: New Frontiers in Graph Learning*, 2023.
- [76] M. Diligenti, F. Giannini, S. Fioravanti, C. Graziani, M. Falaschi, and G. Marra, "Enhancing embedding representations of biomedical data using logic knowledge," in *2023 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2023, pp. 1–8.
- [77] P. Andreini, S. Bonechi, G. Ciano, C. Graziani, V. Lachi, N. Nikolouloupoulou, M. Bianchini, and F. Scarselli, "Multi-stage synthetic image generation for the semantic segmentation of medical images," in *Artificial Intelligence and Machine Learning for Healthcare: Vol. 1: Image and Data Analytics*. Springer, 2022, pp. 79–104.
- [78] G. Giacomini, C. Graziani, V. Lachi, P. Bongini, N. Pancino, M. Bianchini, D. Chiarugi, A. Valleriani, and P. Andreini, "A neural network approach for the analysis of reproducible ribo-seq profiles," *Algorithms*, vol. 15, no. 8, p. 274, 2022.
- [79] E. Stefanescu, N. Pancino, C. Graziani, V. Lachi, M. Sampoli, G. Dimitri, A. Bargagli, D. Zanca, M. Bianchini, D. Mureşanu *et al.*, "Blinking rate comparison between patients with chronic pain and parkinson’s disease," *EUROPEAN JOURNAL OF NEUROLOGY*, vol. 29, pp. 669–669, 2022.
- [80] N. Pancino, C. Graziani, V. Lachi, M. L. Sampoli, E. Ştefănescu, M. Bianchini, and G. M. Dimitri, "A mixed statistical and machine learning approach for the analysis of multimodal trail making test data," *Mathematics*, vol. 9, no. 24, p. 3159, 2021.
- [81] P. Andreini, G. Ciano, S. Bonechi, C. Graziani, V. Lachi, A. Mecocci, A. Sodi, F. Scarselli, and M. Bianchini, "A two-stage gan for high-resolution retinal image generation and segmentation," *Electronics*, vol. 11, no. 1, p. 60, 2021.
- [82] P. Bongini, N. Pancino, V. Lachi, C. Graziani, G. Giacomini, P. Andreini, and M. Bianchini, "Point-wise ribosome translation speed prediction with recurrent neural networks," *Mathematics*, vol. 12, no. 3, p. 465, 2024.
- [83] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 1, pp. 249–270, 2020.

- [84] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.
- [85] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI open*, vol. 1, pp. 57–81, 2020.
- [86] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International conference on machine learning*. PMLR, 2017, pp. 1263–1272.
- [87] V. Garg, S. Jegelka, and T. Jaakkola, "Generalization and representational limits of graph neural networks," in *International Conference on Machine Learning*. PMLR, 2020, pp. 3419–3430.
- [88] G. A. D’Inverno, M. Bianchini, M. L. Sampoli, and F. Scarselli, "On the approximation capability of gnns in node classification/regression tasks," 2023.
- [89] N. M. Kriege, "Weisfeiler and leman go walking: Random walk kernels revisited," in *Advances in Neural Information Processing Systems*, 2022.
- [90] N. Immerman and E. Lander, *Describing graphs: A first-order approach to graph canonization*. Springer, 1990.
- [91] N. T. Huang and S. Villar, "A short tutorial on the weisfeiler-lehman test and its variants," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 8533–8537.
- [92] M. Grohe and M. Otto, "Pebble games and linear equations," *The Journal of Symbolic Logic*, vol. 80, no. 3, pp. 797–844, 2015.
- [93] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "Computational capabilities of graph neural networks," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 81–102, 2008.
- [94] S. Deng, H. Rangwala, and Y. Ning, "Learning dynamic context graphs for predicting social events," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 1007–1016.
- [95] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan, "Inductive representation learning on temporal graphs," *arXiv preprint arXiv:2002.07962*, 2020.
- [96] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein, "Temporal graph networks for deep learning on dynamic graphs," *arXiv preprint arXiv:2006.10637*, 2020.
- [97] R. Trivedi, H. Dai, Y. Wang, and L. Song, "Know-evolve: Deep temporal reasoning for dynamic knowledge graphs," in *international conference on machine learning*. PMLR, 2017, pp. 3462–3471.
- [98] J. Wu, M. Cao, J. C. K. Cheung, and W. L. Hamilton, "Temp: Temporal message passing for temporal knowledge graph completion," *arXiv preprint arXiv:2010.03526*, 2020.
- [99] J. Skarding, B. Gabrys, and K. Musial, "Foundations and modeling of dynamic networks using dynamic graph neural networks: A survey," *IEEE Access*, vol. 9, pp. 79 143–79 168, 2021.
- [100] J. Thomas, A. Moallem-Oureh, S. Beddar-Wiesing, and C. Holzhüter, "Graph neural networks designed for different graph types: A survey," *Transactions on Machine Learning Research*, 2022.

- [101] C. D. Barros, M. R. Mendonça, A. B. Vieira, and A. Ziviani, “A survey on embedding dynamic graphs,” *ACM Computing Surveys (CSUR)*, vol. 55, no. 1, pp. 1–37, 2021.
- [102] A. Longa, V. Lachi, G. Santin, M. Bianchini, B. Lepri, P. Lio, F. Scarselli, and A. Passerini, “Graph neural networks for temporal graphs: State of the art, open challenges, and opportunities,” *arXiv preprint arXiv:2302.01018*, 2023.
- [103] G. Xue, M. Zhong, J. Li, J. Chen, C. Zhai, and R. Kong, “Dynamic network embedding survey,” *Neurocomputing*, vol. 472, pp. 212–223, 2022.
- [104] J. M. Thomas, S. Beddar-Wiesing, A. Moallem-Oureh, and R. Nather, “A note on the modeling power of different graph types,” *arXiv preprint arXiv:2109.10708*, 2021.
- [105] S. M. Kazemi, R. Goel, K. Jain, I. Kobyzev, A. Sethi, P. Forsyth, and P. Poupart, “Representation learning for dynamic graphs: A survey,” *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 2648–2720, 2020.
- [106] B. Hammer, “On the approximation capability of recurrent neural networks,” *Neurocomputing*, vol. 31, no. 1-4, pp. 107–123, 2000.
- [107] S. Kiefer and B. D. McKay, “The iteration number of colour refinement,” *arXiv preprint arXiv:2005.10182*, 2020.
- [108] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, “Structured sequence modeling with graph convolutional recurrent networks,” in *Neural Information Processing: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13-16, 2018, Proceedings, Part I 25*. Springer, 2018, pp. 362–373.
- [109] A. Narayan and P. H. Roe, “Learning graph dynamics using deep neural networks,” *Ifac-Papersonline*, vol. 51, no. 2, pp. 433–438, 2018.
- [110] M. Niepert, M. Ahmed, and K. Kutzkov, “Learning convolutional neural networks for graphs,” in *International conference on machine learning*. PMLR, 2016, pp. 2014–2023.
- [111] A. Taheri, K. Gimpel, and T. Berger-Wolf, “Learning to represent the evolution of dynamic graphs with recurrent models,” in *Companion proceedings of the 2019 world wide web conference*, 2019, pp. 301–307.
- [112] M. Gehrke and S. van Gool, “Topological duality for distributive lattices, and applications,” *arXiv preprint arXiv:2203.03286*, 2022.
- [113] M. Grohe, *Descriptive complexity, canonisation, and definable graph structure theory*. Cambridge University Press, 2017, vol. 47.
- [114] E. Thiede, W. Zhou, and R. Kondor, “Autobahn: Automorphism-based graph neural nets,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 29 922–29 934, 2021.
- [115] M. Zhang and P. Li, “Nested graph neural networks,” in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34, 2021, pp. 15 734–15 747.
- [116] C. Bodnar, F. Frasca, N. Otter, Y. Wang, P. Lio, G. F. Montufar, and M. Bronstein, “Weisfeiler and lehman go cellular: Cw networks,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 2625–2640, 2021.
- [117] F. M. Bianchi and V. Lachi, “The expressive power of pooling in graph neural networks,” *Advances in neural information processing systems*, vol. 36, 2024.
- [118] C. Bodnar, F. Frasca, Y. Wang, N. Otter, G. F. Montufar, P. Lio, and M. Bronstein, “Weisfeiler and lehman go topological: Message passing simplicial networks,” in *International Conference on Machine Learning*. PMLR, 2021, pp. 1026–1037.

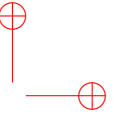
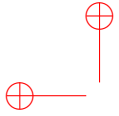
- [119] B. Bevilacqua, F. Frasca, D. Lim, B. Srinivasan, C. Cai, G. Balamurugan, M. M. Bronstein, and H. Maron, “Equivariant subgraph aggregation networks,” *arXiv preprint arXiv:2110.02910*, 2021.
- [120] G. Bouritsas, F. Frasca, S. Zafeiriou, and M. M. Bronstein, “Improving graph neural network expressivity via subgraph isomorphism counting,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 1, pp. 657–668, 2022.
- [121] R. Abboud, R. Dimitrov, and I. I. Ceylan, “Shortest path networks for graph property prediction,” in *Learning on Graphs Conference*. PMLR, 2022, pp. 5–1.
- [122] L. Kong, Y. Chen, and M. Zhang, “Geodesic graph neural network for efficient graph representation learning,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 5896–5909, 2022.
- [123] P. Li, Y. Wang, H. Wang, and J. Leskovec, “Distance encoding: Design provably more powerful neural networks for graph representation learning,” in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 4465–4478.
- [124] Y. Ding, A. Orvieto, B. He, and T. Hofmann, “Recurrent distance-encoding neural networks for graph representation learning,” *arXiv preprint arXiv:2312.01538*, 2023.
- [125] C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, and T.-Y. Liu, “Do transformers really perform badly for graph representation?” *Advances in Neural Information Processing Systems*, vol. 34, pp. 28 877–28 888, 2021.
- [126] G. Nikolentzos, G. Dasoulas, and M. Vazirgiannis, “k-hop graph neural networks,” *Neural Networks*, vol. 130, pp. 195–205, 2020.
- [127] J. Feng, Y. Chen, F. Li, A. Sarkar, and M. Zhang, “How powerful are k-hop message passing graph neural networks,” in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 4776–4790.
- [128] Y. Sun, H. Deng, Y. Yang, C. Wang, J. Xu, R. Huang, L. Cao, Y. Wang, and L. Chen, “Beyond homophily: Structure-aware path aggregation graph neural network,” in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, L. D. Raedt, Ed. International Joint Conferences on Artificial Intelligence Organization, 7 2022, pp. 2233–2240, main Track.
- [129] Q. Truong and P. Chin, “Generalizing topological graph neural networks with paths,” *arXiv preprint arXiv:2308.06838*, 2023.
- [130] G. Michel, G. Nikolentzos, J. F. Lutzeyer, and M. Vazirgiannis, “Path neural networks: Expressive and accurate graph neural networks,” in *International Conference on Machine Learning*. PMLR, 2023, pp. 24 737–24 755.
- [131] C. Bodnar, F. Frasca, N. Otter, Y. G. Wang, P. Liò, G. Montúfar, and M. Bronstein, “Weisfeiler and Lehman go cellular: CW networks,” in *NeurIPS*, 2021.
- [132] J. You, J. M. Gomes-Selman, R. Ying, and J. Leskovec, “Identity-aware graph neural networks,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, 2021, pp. 10 737–10 745.
- [133] C. Qian, G. Rattan, F. Geerts, M. Niepert, and C. Morris, “Ordered subgraph aggregation networks,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 21 030–21 045, 2022.

- [134] C. Morris, G. Rattan, and P. Mutzel, “Weisfeiler and leman go sparse: Towards scalable higher-order graph embeddings,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 824–21 840, 2020.
- [135] C. Morris, G. Rattan, S. Kiefer, and S. Ravanbakhsh, “Speqnets: Sparsity-aware permutation-equivariant graph networks,” in *International Conference on Machine Learning*. PMLR, 2022, pp. 16 017–16 042.
- [136] B. Zhang, G. Feng, Y. Du, D. He, and L. Wang, “A complete expressiveness hierarchy for subgraph gnns via subgraph weisfeiler-lehman tests,” *arXiv preprint arXiv:2302.07090*, 2023.
- [137] F. Frasca, B. Bevilacqua, M. Bronstein, and H. Maron, “Understanding and extending subgraph gnns by rethinking their symmetries,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 31 376–31 390, 2022.
- [138] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann, “TUDataset: A collection of benchmark datasets for learning with graphs,” in *ICML Workshop on Graph Representation Learning and Beyond*, 2020.
- [139] D. Neuen, “Homomorphism-distinguishing closedness for graphs of bounded tree-width,” 2023.
- [140] S. Kiefer and D. Neuen, “The power of the weisfeiler–leman algorithm to decompose graphs,” *SIAM Journal on Discrete Mathematics*, vol. 36, no. 1, pp. 252–298, 2022.
- [141] Y. Huang, X. Peng, J. Ma, and M. Zhang, “Boosting the cycle counting power of graph neural networks with \mathbb{I}^2 -GNNs,” in *The Eleventh International Conference on Learning Representations*, 2022.
- [142] B. Zhang, J. Gai, Y. Du, Q. Ye, D. He, and L. Wang, “Beyond weisfeiler-lehman: A quantitative framework for gnn expressiveness,” 2024.
- [143] J. Feng, L. Kong, H. Liu, D. Tao, F. Li, M. Zhang, and Y. Chen, “Towards arbitrarily expressive gnns in $O(n^2)$ space by rethinking folklore weisfeiler-lehman,” *arXiv preprint arXiv:2306.03266*, 2023.
- [144] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [145] R. Abboud, I. I. Ceylan, M. Grohe, and T. Lukasiewicz, “The surprising power of graph neural networks with random node initialization,” in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 8 2021, pp. 2112–2118.
- [146] M. Balcilar, P. Héroux, B. Gauzere, P. Vasseur, S. Adam, and P. Honeine, “Breaking the limits of message passing graph neural networks,” in *International Conference on Machine Learning*. PMLR, 2021, pp. 599–608.
- [147] R. Murphy, B. Srinivasan, V. Rao, and B. Ribeiro, “Relational pooling for graph representations,” in *ICML*, 2019.
- [148] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik, “Automatic chemical design using a data-driven continuous representation of molecules,” *ACS Central Science*, 2018.
- [149] T. Sterling and J. J. Irwin, “ZINC 15 – ligand discovery for everyone,” *Journal of Chemical Information and Modeling*, 2015.

- [150] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.
- [151] B. Bevilacqua, F. Frasca, D. Lim, B. Srinivasan, C. Cai, G. Balamurugan, M. Bronstein, and H. Maron, "Equivariant subgraph aggregation networks," in *ICLR*, 2021.
- [152] R. Cao, "mTOR signaling, translational control, and the circadian clock," *Frontiers in genetics*, vol. 9, p. 367, 2018.
- [153] N. T. Ingolia, G. A. Brar, S. Rouskin, A. M. McGeachy, and J. S. Weissman, "The ribosome profiling strategy for monitoring translation in vivo by deep sequencing of ribosome-protected mRNA fragments," *Nature protocols*, vol. 7, no. 8, pp. 1534–1550, 2012.
- [154] C. Sin, D. Chiarugi, and A. Valleriani, "Quantitative assessment of ribosome drop-off in *e. coli*," *Nucleic acids research*, vol. 44, no. 6, pp. 2528–2537, 2016.
- [155] A. Valleriani and D. Chiarugi, "A workbench for the translational control of gene expression," *bioRxiv*, 2020.
- [156] S. K. Archer, N. E. Shirokikh, T. H. Beilharz, and T. Preiss, "Dynamics of ribosome scanning and recycling revealed by translation complex profiling," *Nature*, vol. 535, no. 7613, pp. 570–574, 2016.
- [157] W. Li, G. Qi, and Q. Ji, "Hybrid reasoning in knowledge graphs: Combing symbolic reasoning and statistical reasoning," *Semantic Web*, vol. 11, no. 1, pp. 53–62, 2020.
- [158] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich, "A review of relational machine learning for knowledge graphs: From multi-relational link prediction to automated knowledge graph construction," *CoRR*, vol. abs/1503.00759, 2015. [Online]. Available: <http://arxiv.org/abs/1503.00759>
- [159] Y. Dai, S. Wang, N. N. Xiong, and W. Guo, "A survey on knowledge graph embedding: Approaches, applications and benchmarks," *Electronics*, vol. 9, no. 5, p. 750, 2020.
- [160] S. Zheng, J. Rao, Y. Song, J. Zhang, X. Xiao, E. F. Fang, Y. Yang, and Z. Niu, "Pharmkg: a dedicated knowledge graph benchmark for biomedical data mining," *Briefings in bioinformatics*, vol. 22, no. 4, p. bbaa344, 2021.
- [161] G. Marra, M. Diligenti, and F. Giannini, "Relational reasoning networks," *arXiv preprint arXiv:2106.00393*, 2021.
- [162] L. A. Galárraga, C. Teflioudi, K. Hose, and F. Suchanek, "Amie: association rule mining under incomplete evidence in ontological knowledge bases," in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 413–422.
- [163] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.
- [164] C.-P. Lim, A. Vaidya, Y.-W. Chen, T. Jain, and L. C. Jain, *Artificial Intelligence and Machine Learning for Healthcare: Vol. 1: Image and Data Analytics*. Springer Nature, 2022, vol. 228.
- [165] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017.
- [166] J. Staal, M. D. Abramoff, M. Niemeijer, M. A. Viergever, and B. Van Ginneken, "Ridge-based vessel segmentation in color images of the retina," *IEEE transactions on medical imaging*, vol. 23, no. 4, pp. 501–509, 2004.

- [167] M. M. Fraz, P. Remagnino, A. Hoppe, B. Uyyanonvara, A. R. Rudnicka, C. G. Owen, and S. A. Barman, "An ensemble classification-based approach applied to retinal blood vessel segmentation," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 9, pp. 2538–2548, 2012.
- [168] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2017.
- [169] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.
- [170] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2881–2890.
- [171] G. Chéron, I. Laptev, and C. Schmid, "P-CNN: Pose-based CNN features for action recognition," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3218–3226.
- [172] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2961–2969.
- [173] T.-C. Huynh, "Vision-based autonomous bolt-looseness detection method for splice connections: Design, lab-scale evaluation, and field application," *Automation in Construction*, vol. 124, p. 103591, 2021.
- [174] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.
- [175] K. Holmqvist, M. Nyström, R. Andersson, R. Dewhurst, H. Jarodzka, and J. Van de Weijer, *Eye tracking: A comprehensive guide to methods and measures*. oup Oxford, 2011.
- [176] G. Veneri, E. Pretelegiani, F. Rosini, P. Federighi, A. Federico, and A. Rufa, "Evaluating the human ongoing visual search performance by eye tracking application and sequencing tests," *Computer methods and programs in biomedicine*, vol. 107, no. 3, pp. 468–477, 2012.
- [177] W. H. Kruskal and W. A. Wallis, "Use of ranks in one-criterion variance analysis," *Journal of the American statistical Association*, vol. 47, no. 260, pp. 583–621, 1952.
- [178] A. Vargha and H. D. Delaney, "The kruskal-wallis test and stochastic homogeneity," *Journal of Educational and behavioral Statistics*, vol. 23, no. 2, pp. 170–192, 1998.
- [179] B. G. Amidan, T. A. Ferryman, and S. K. Cooley, "Data outlier detection using the chebyshev theorem," in *2005 IEEE Aerospace Conference*. IEEE, 2005, pp. 3814–3819.
- [180] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical image computing and computer-assisted intervention—MICCAI 2015: 18th international conference, Munich, Germany, October 5–9, 2015, proceedings, part III 18*. Springer, 2015, pp. 234–241.
- [181] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.

- [182] F. Geerts, “Walk message passing neural networks and second-order graph neural networks,” *CoRR*, vol. abs/2006.09499, 2020.
- [183] H. Maron, H. Ben-Hamu, N. Shamir, and Y. Lipman, “Invariant and equivariant graph networks,” in *7th International Conference on Learning Representations*, 2019.
- [184] J. You, R. Ying, and J. Leskovec, “Position-aware graph neural networks,” in *International conference on machine learning*. PMLR, 2019, pp. 7134–7143.
- [185] D. Lim, J. D. Robinson, L. Zhao, T. Smidt, S. Sra, H. Maron, and S. Jegelka, “Sign and basis invariant networks for spectral graph representation learning,” in *International Conference on Learning Representations*, 2022.
- [186] Z. Dong, M. Zhang, P. Payne, M. A. Province, C. Cruchaga, T. Zhao, F. Li, and Y. Chen, “Rethinking the power of graph canonization in graph representation learning with stability,” in *The Twelfth International Conference on Learning Representations*, 2023.
- [187] E. Chien, J. Peng, P. Li, and O. Milenkovic, “Adaptive universal generalized pagerank graph neural network,” in *9th International Conference on Learning Representations*, 2021.
- [188] S. Abu-El-Haija, B. Perozzi, A. Kapoor, N. Alipourfard, K. Lerman, H. Harutyunyan, G. Ver Steeg, and A. Galstyan, “Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing,” in *International Conference on Machine Learning*. PMLR, 2019, pp. 21–29.
- [189] G. Wang, R. Ying, J. Huang, and J. Leskovec, “Multi-hop attention graph neural networks,” in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, August 2021*, 2021, pp. 3089–3096.
- [190] P. A. Papp and R. Wattenhofer, “A theoretical comparison of graph neural network extensions,” in *International Conference on Machine Learning*. PMLR, 2022, pp. 17323–17345.
- [191] N. Robertson and P. D. Seymour, “Graph minors. ii. algorithmic aspects of tree-width,” *Journal of algorithms*, vol. 7, no. 3, pp. 309–322, 1986.
- [192] V. Balakrishnan, *Schaum’s Outline of Graph Theory: Including Hundreds of Solved Problems*. McGraw Hill Professional, 1997.
- [193] N. Biggs, E. K. Lloyd, and R. J. Wilson, *Graph Theory, 1736-1936*. Oxford University Press, 1986.
- [194] R. W. Robinson and N. C. Wormald, “Almost all regular graphs are hamiltonian,” *Random Structures & Algorithms*, vol. 5, no. 2, pp. 363–374, 1994.
- [195] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, “Pytorch: An imperative style, high-performance deep learning library,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [196] M. Fey and J. E. Lenssen, “Fast graph representation learning with PyTorch Geometric,” in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.



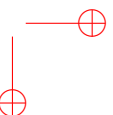
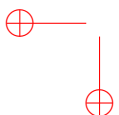
Appendix A

Other Works: Statistical and Machine Learning Approaches for Biomedical Applications

This chapter provides an overview of other research activities that have characterized my PhD but are not closely related to the thesis topic. I would like to start with two papers which are related to my Master's Degree thesis. The common research question is: what is driving the irregular dynamics of ribosomes during the translation phase?

A Neural Network Approach for the Analysis of Reproducible Ribo-Seq Profiles [78]

Ribosomes perform protein synthesis from mRNA templates by a highly regulated process called translation. Translation control plays a key role in the regulation of gene expression, both in physiological and pathological conditions [152]. The advent of high-throughput methods to measure the levels of gene expression has revealed the implications of multiple factors that might impact the rate at which an mRNA is translated. In recent years, the Ribosome profiling technique (Ribo-seq) has emerged as a powerful method for globally monitoring the translation process in vivo at single nucleotide resolution [153]. Based on deep sequencing of mRNA fragments, Ribo-seq allows to obtain profiles that reflect the time spent by ribosomes in translating each part of an open reading frame. Unfortunately, the profiles produced by this method can vary significantly in different experimental setups, being characterized by poor reproducibility [154]. To address this problem, we have employed a statistical method for the identification of highly reproducible Ribo-seq profiles, which was tested on a set of *E. coli* genes. In particular, inspired by the seminal work proposed in [155], we perform a novel analysis procedure for Ribo-seq data that allows us to identify the reproducible Ribo-seq profiles emerging from the comparison of independent Ribo-seq experiments performed in different laboratories under the same conditions. These significantly reproducible profiles are then collected into a library of consensus sequences, in which sub-regions characterized by different translation speeds can be isolated. State-of-the-art artificial neural network models have been used to validate the quality of the produced sequences. Moreover, new insights into the dynamics of ribosome translation have been provided through a statistical analysis of the obtained sequences. The analysis showed a statistically significant difference in the nucleotide composition between sub-sequences characterized by different translation speeds. The combinations of these experiments allowed us to discover that the translation speed is modulated both by the nucleotide composition of the sequences and by the order in which they appear within each sequence. Our code is available at <https://github.com/pandrein/Ribo-Seq-analysis>.



Point-wise Ribosome Translation Speed Prediction with Recurrent Neural Networks [82]

Escherichia Coli is a benchmark organism and it has been deeply studied by the scientific community for decades, obtaining a vast amount of metabolic and genetic data. Among these data, estimates of the translation speed of ribosomes over its genome are available. The molecular process of translating mRNA into proteins is a cornerstone of cellular biology, representing a critical intersection between the genetic code and functional biomolecules. Translation is executed by ribosomes, which are sophisticated molecular complexes composed of RNA and protein components. These complexes function as the sites of protein synthesis, interpreting the genetic information encoded in mRNA sequences and assembling corresponding amino acids into polypeptide chains, which subsequently fold into functional proteins, essential for cell life. Translation is not merely a mechanical process but is intricately regulated, playing a pivotal role in the control of gene expression. This regulation is essential for maintaining cellular homeostasis, enabling cells to adapt protein production to suit specific tissue requirements and to respond to a wide variety of internal and external stimuli. The fidelity and efficiency of translation are critical, and disruptions in these processes are frequently implicated in disease mechanisms [152], highlighting the importance of understanding translation at a molecular level. The complexity of translation extends beyond the ribosome-mRNA interactions. The ribosome itself is a dynamic entity, capable of various interactions with the mRNA molecule, the emerging peptide chain, and external molecular factors [156]. These interactions are not merely mechanical but are intricately regulated, contributing to the efficiency of protein synthesis. Recognizing these interactions is crucial for a comprehensive understanding of translation and its role in cellular function and pathology. In order to understand if there exist biological signals to suggest ribosomes how to move on the mRNA strand, we decided to process the mRNA sequences with machine learning models. The objective is to predict the translation speed of each nucleotide, codon, or amino acid inside a sequence. The models are trained, validated, and tested on a dataset of *E.Coli* Open Reading Frames (ORFs), obtained from a consensus pool of 9 source datasets. Once trained and successfully validated, the model can then be exploited to predict the translation speed of all the *E.Coli* ORFs for which the consensus threshold was not reached, thus marking the uncertainty in determining the translation speed with traditional methods. On one hand, this method can help build models for reliable predictions of mRNA translation speed, reducing the future need for more costly Ribo-Seq experiments. On the other hand, ablation studies and attention mechanisms can help explain the models' decisions, thus identifying the factors that determine the speed in nature, and their importance.

In the next work, we made use of a *neuro-symbolic* approach to refine the representations of the entries of a large biomedical knowledge graph. In neuro-symbolic methods, symbolic knowledge (in this case first-order logic rules) is integrated into a neural component and can provide useful additional information to improve the performance [157].

Enhancing Embedding Representations of Biomedical Data using Logic Knowledge [76]

Knowledge Graph Embeddings (KGE) have become a quite popular class of models specifically devised to deal with ontologies and graph structure data, as they can implicitly encode statistical dependencies between entities and relations in a latent space [158, 159]. KGE techniques are particularly effective for the biomedical domain, where it is quite common to deal with large knowledge graphs underlying complex interactions between biological and chemical objects. Recently in the literature, the PharmKG dataset has been proposed as one of the most challenging knowledge graph biomedical benchmarks, with hundreds of thousands of relational facts between genes, diseases, and chemicals [160]. Although KGEs can scale to very large relational domains, they generally fail to represent more complex relational dependencies between facts, therefore we claim that logic rules can constitute a valuable source of additional knowledge to establish complex interconnections among these entities. However, given the remarkable dimension of this benchmark, successfully applying methodologies that combine KGEs and logic rules is very challenging and, according to the authors' knowledge, this is the first work accomplishing this task. To this end, we adopt Relational Reasoning Network (R2N), a recently proposed neural-symbolic approach [161] showing promising results on knowledge graph completion tasks. An R2N uses the available logic rules to build a neural architecture that reasons over KGE latent representations. In the experiments, we show that our approach is able to significantly improve the current state-of-the-art on the PharmKG dataset. Finally, we show how it is possible to employ automatic rule mining techniques to enrich knowledge graph completion tasks [162], and we provide an in-depth ablation study to evaluate the performance of R2Ns according to different settings of the rule miner.

Other than graphs and sequences, I worked with images. In particular, I focused on two powerful techniques, namely, image generation and semantic segmentation. The standard model to generate new images from existing data is the Generative Adversarial Network (GAN) [163]. On the other hand, image segmentation is the task of classifying portions of the image as belonging to a particular semantic area of the picture (for example main subject/ background). This allows us to simplify and/or change the representation of an image into something more meaningful and easier to analyze. This is crucial in applications such as medical imaging, where precise analysis can aid in diagnosis and treatment planning. In particular, we targeted retinal fundus images. The paper on this topic has been included also in a chapter of the book *Artificial Intelligence and Machine Learning for Healthcare* [164]. We provide a summary of both versions.

A Two-Stage GAN for High-Resolution Retinal Image Generation and Segmentation[81]

In this paper, we use Generative Adversarial Networks (GANs) to synthesize high-quality retinal images along with the corresponding semantic label-maps, instead of real images during training of a segmentation network. Different from other previous proposals, we employ a two-step approach: first, a progressively growing GAN [165] is trained to generate the semantic label-maps, which describes the blood vessel structure (i.e., the vasculature); second, an image-to-image translation approach is used to obtain realistic retinal images from the generated vasculature. The adoption of a two-stage process simplifies the generation task so that the network training requires fewer images with consequent lower memory usage. Moreover, learning is effective, and with only a handful of training samples, our approach generates realistic high-resolution images, which can be successfully used to enlarge small datasets such as DRIVE [166] and CHASE-DB1 [167]. Comparable results were obtained by employing only synthetic images in place of real data during training. The practical viability of the proposed approach was demonstrated on two well-established benchmark sets for retinal vessel segmentation—both containing a very small number of training samples—obtaining better performance with respect to state-of-the-art techniques.

Multi-stage Synthetic Image Generation for the Semantic Segmentation of Medical Images [77]

Recently, deep learning methods have had a tremendous impact on computer vision applications, from image classification and semantic segmentation [168, 169, 170] to object detection and face recognition [171, 172, 173, 174]. Nevertheless, the training of state-of-the-art neural network models is usually based on the availability of large sets of supervised data. Indeed, deep neural networks have a huge number of parameters that, to be properly trained, require a fairly large dataset of supervised examples. This problem is particularly relevant in the medical field due to privacy issues and the high cost of image tagging by medical experts. In this chapter, we present a new approach that allows to reduce this limitation by generating synthetic images with their corresponding supervision. In particular, this approach can be applied in semantic segmentation, where the generated images (and label-maps) can be used to augment real datasets during network training. The main characteristic of our method, differently from other existing techniques, lies in the generation procedure carried out in multiple steps, based on the intuition that, by splitting the procedure in multiple phases, the overall generation task is simplified. The effectiveness of the proposed multi-stage approach has been evaluated on two different domains, retinal fundus and chest X-ray images. In both domains, the multi-stage approach has been compared with the single-stage generation procedure. The results suggest that generating images in multiple steps is more effective and computationally cheaper, yet allows high-resolution, realistic images to be used for training deep networks.

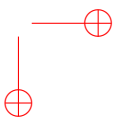
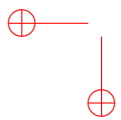
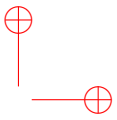
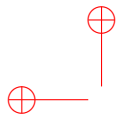
The Trail Making Test (TMT) is used in clinical practice as a neuropsychological assessment of visual attention and task switching. The test investigates the subject's attentive abilities and the capability to quickly switch from a numerical to an alphabetical visual stimulus. We exploited an oculomotor-based methodology, called eye-tracking, to study cognitive impairments in patients affected by chronic pain and extrapyramidal syndrome, such as Parkinson's disease. Here we briefly present the main paper and a spin-off published in the *European journal of neurology*, focusing on one of the three indicators detected in [80]: the spontaneous blinking rate.

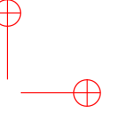
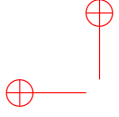
A Mixed Statistical and Machine Learning Approach for the Analysis of Multimodal Trail Making Test Data [80]

Eye-tracking can offer a novel clinical practice and a non-invasive tool to detect neuropathological syndromes [175]. In this paper, we show some analysis of data obtained from the visual sequential search test [176]. Indeed, such a test can be used to evaluate the capacity to look at objects in a specific order, and its successful execution requires the optimization of the perceptual resources of foveal and extrafoveal vision. The main objective of this work is to detect if some patterns can be found within the data to discern among people with chronic pain, extrapyramidal patients, and healthy controls. We employed statistical tests [177, 178, 179] to evaluate differences among groups, considering three novel indicators: blinking rate, average blinking duration, and maximum pupil size variation. Additionally, to divide the three patient groups based on scan-path images—which appear very noisy and all similar to each other—we applied deep learning techniques [180] to embed them into a larger transformed space. We then applied a clustering approach [181] to correctly detect and classify the three cohorts. Preliminary experiments show promising results.

Blinking Rate Comparison Between Patients with Chronic Pain and Parkinson's Disease [79]

Blinking can be spontaneous, voluntary, or reflex. Spontaneous blink rate (SBR) is strictly related to dopamine levels in the central nervous system and is considered a reliable noninvasive biomarker of central dopaminergic activity. Reduced spontaneous blinking rate is a common finding in Parkinson's disease and other parkinsonian syndromes suggesting involvement of fronto-striatal system and a reduced Dopamine tone. Recent evidence indicates a possible relationship between the central dopaminergic system and pain modulation in both animal and human studies. A subpopulation of dopaminergic neurons within the ventrolateral periaqueductal grey (PAG), which is included in the pain modulatory network, projects to brain regions known to be involved in pain modulation. D1 and D2 receptors are expressed in the PAG and seem to have an antinociception activity. The aim of this study is to investigate changes of SBR in patients with Chronic Pain (CP) compared to normal subjects and PD.





Appendix B

Appendix

The present appendix collects all the proofs that were not included in the main text of Part II of the thesis (Chapter 3 and 4). We provide first some additional preliminaries.

B.1 Additional Preliminaries

Definition B.1 (Different Graph Types). The elementary graphs are defined in [104] as follows.

- A **directed graph (digraph)** is a tuple $G = (\mathcal{V}, \mathcal{E})$ containing a set of nodes $\mathcal{V} \subseteq \mathbb{N}$ and a set of directed edges given as tuples $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$.
- A **(generalized) directed hypergraph** is a tuple $G = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V} \subseteq \mathbb{N}$ and hyperedges $\mathcal{E} \subseteq \{(x, f_i)_i \mid x \subseteq \mathcal{V}, f_i : x \rightarrow \mathbb{N}_0\}$ that include a numbering map f_i for the i -th edge $(x, f)_i$ which indicates the order of the nodes in the (generalized) directed hyperedge.

An elementary graph $G = (\mathcal{V}, \mathcal{E})$ is called

- **undirected** if the directions of the edges are irrelevant, i.e.,
 - for directed graphs: if $(u, v) \in \mathcal{E}$ whenever $(v, u) \in \mathcal{E}$ for $u, v \in \mathcal{V}$.
 - for directed hypergraphs: if $f_i : x \rightarrow 0$ for all $(x, f_i)_i \in \mathcal{E}^1$.
- **multigraph** if the node or edge sets are multisets.
- **heterogeneous** if the nodes or edges can have different types. I.e., the node set is determined by $\mathcal{V} \subseteq \mathbb{N} \times \mathcal{S}$ with a node type set \mathcal{S} and thus, a node $(v, s) \in \mathcal{V}$ is given by the node v itself and its type s . The edges can be extended by a set \mathcal{R} that describes their types, to $(e, r) \forall e \in \mathcal{E}$ of edge type $r \in \mathcal{R}$. Otherwise, the graph is called **homogeneous**.
- **attributed** if the nodes \mathcal{V} or edges \mathcal{E} are equipped with node or edge attributes², formally given by a node and edge attribute function, respectively, i.e. $\alpha : \mathcal{V} \rightarrow \mathcal{A}$ and $\omega : \mathcal{E} \rightarrow \mathcal{W}$, where \mathcal{A} and \mathcal{W} are arbitrary attribute sets.

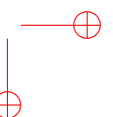
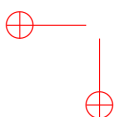
[104] claims that there exists a bijective transformation from any arbitrary graph type defined in Def. B.1 to the SAUHG. They result from concatenating transformations for single graph properties and are sketched in the following.

Graph Type Transformations. Given a graph of arbitrary type, the concatenation of suitable transformations from the following list leads to the SAUHG type.

- **Hypergraph \rightarrow Simple Graph:** Transform undirected hyperedges to fully connected subgraphs and directed edges to chained directed bipartite subgraphs given the hyperedge direction.

¹ $f_i(x) = 0$ encodes that x is an undirected hyperedge

²In some literature attributes are also called features.



- **Multigraph** \rightarrow **Simple Graph**: Encode multiple nodes or edges in an additional counter each and, i.a., concatenate the corresponding attributes in a vector.
- **Dynamic** \rightarrow **Static**: Encode the dynamical behaviour of the graph into a time series of node and edge attributes accordingly.
- **Unattributed** \rightarrow **Attributed**: Add empty attributes to each node or edge.
- **Directed** \rightarrow **Undirected**: Replace all directed edges with undirected ones and add the direction as encoded additional information into the edge attributes.
- **Heterogenous** \rightarrow **Homogeneous**: Encode the different node or edge types in an additional dimension of the node or edge attribute, respectively.

B.2 Universal Approximation Capabilities

Here are collected the proofs of Lemmas, Propositions, and main Theorems on approximation capabilities of AGNNs and DGNNs.

B.2.1 Proof of Lemma 3.1

Statement. Consider $G = (\mathcal{V}, \mathcal{E}, \alpha, \omega)$ with nodes $u, v \in \mathcal{V}$ and corresponding attributes α_u, α_v . Then it holds

$$\forall d \in \mathbb{N}_0 : \mathbf{T}_u^d = \mathbf{T}_v^d \iff \mathbf{c}_u^{(d)} = \mathbf{c}_v^{(d)} \quad (\text{B.1})$$

Proof. The proof is carried out by induction on d , which represents both the depth of the unfolding trees and the iteration step in the WL coloring.

$d = 0$: It holds

$$\begin{aligned} \mathbf{T}_u^0 &= \text{Tree}(\alpha_u) = \text{Tree}(\alpha_v) = \mathbf{T}_v^0 \\ \iff \alpha_u &= \alpha_v \iff \mathbf{c}_u^{(0)} = \text{Hash}(\alpha_u) = \text{Hash}(\alpha_v) = \mathbf{c}_v^{(0)}. \end{aligned}$$

because of the injectivity of the **Hash** function.

$d > 0$: Suppose that Eq. (B.1) holds for $d - 1$, and prove that it holds also for d .

- By definition, $\mathbf{T}_u^d = \mathbf{T}_v^d$ is equivalent to

$$\begin{cases} \mathbf{T}_u^{d-1} = \mathbf{T}_v^{d-1} & \text{and} \\ \text{Tree}(\alpha_u, \{\{(\omega_{(n,u)}, \mathbf{T}_n^{d-1})\}_{n \in \mathcal{N}(u)}\}) = \text{Tree}(\alpha_v, \{\{(\omega_{(n',v)}, \mathbf{T}_{n'}^{d-1})\}_{n' \in \mathcal{N}(v)}\}). \end{cases} \quad (\text{B.2})$$

- Applying the induction hypothesis, it holds that

$$\mathbf{T}_u^{d-1} = \mathbf{T}_v^{d-1} \iff \mathbf{c}_u^{(d-1)} = \mathbf{c}_v^{(d-1)}. \quad (\text{B.3})$$

- Eq. (B.2) is equivalent to the following:

$$\alpha_u = \alpha_v, \quad \{\{(\omega_{(n,u)}, \mathbf{T}_n^{d-1})\}_{n \in \mathcal{N}(u)}\} = \{\{(\omega_{(n',v)}, \mathbf{T}_{n'}^{d-1})\}_{n' \in \mathcal{N}(v)}\}$$

That is equivalent to:

$$\begin{cases} \omega_{(u,n)} = \omega_{(v,n')} & \forall n \in \mathcal{N}(u), n' \in \mathcal{N}(v) \text{ and} \\ \mathbf{T}_n^{d-1} = \mathbf{T}_{n'}^{d-1} & \forall n \in \mathcal{N}(u), n' \in \mathcal{N}(v). \end{cases} \quad (\text{B.4})$$

- By the induction hypothesis, Eq. (B.4) is equivalent to

$$\{\{\mathbf{c}_n^{(d-1)} \mid n \in \mathcal{N}(u)\}\} = \{\{\mathbf{c}_{n'}^{(d-1)} \mid n' \in \mathcal{N}(v)\}\}.$$

- Putting together Eq. (B.3), (B.4), and the fact that the **Hash** function is bijective, we obtain:

$$\begin{aligned} & \text{Hash}\left(\left(\mathbf{c}_u^{(d-1)}, \{\{\mathbf{c}_n^{(d-1)} \mid n \in \mathcal{N}(u)\}\}, \{\{\omega_{(n,u)} \mid n \in \mathcal{N}(u)\}\}\right)\right) \\ &= \text{Hash}\left(\left(\mathbf{c}_v^{(d-1)}, \{\{\mathbf{c}_{n'}^{(d-1)} \mid n' \in \mathcal{N}(v)\}\}, \{\{\omega_{(n',v)} \mid n' \in \mathcal{N}(v)\}\}\right)\right) \end{aligned}$$

which, by definition, is equivalent to $\mathbf{c}_u^{(d)} = \mathbf{c}_v^{(d)}$.

□

B.2.2 Proof of Proposition 3.4

Statement. A function f belongs to $\mathcal{F}(\mathcal{G})$ if and only if there exists a function κ defined on trees such that for any graph $G' \in \mathcal{G}$ it holds $f(G', v) = \kappa(\mathbf{T}_v)$, for any node $v \in G'$.

Proof. We prove by showing both equivalence directions:

⇐ If there exists a function κ on attributed unfolding trees such that $f(G', v) = \kappa(\mathbf{T}_v)$ for all $v \in G'$, then $u \sim_{AUT} v$ for $u, v \in G$ implies $f(G', u) = \kappa(\mathbf{T}_u) = \kappa(\mathbf{T}_v) = f(G', v)$.

⇒ If f preserves the attributed unfolding equivalence, then a function κ on the attributed unfolding tree of an arbitrary node v can be defined as $\kappa(\mathbf{T}_v) := f(G', v)$. Then, if \mathbf{T}_u and \mathbf{T}_v are two attributed unfolding trees, $\mathbf{T}_u = \mathbf{T}_v$ implies $f(G', u) = f(G', v)$ and κ is uniquely defined.

□

B.2.3 Proof of Proposition 3.7

Statement. A dynamic system belongs to $\mathcal{F}(\mathcal{D})$ if and only if there exists a function κ defined on attributed trees such that for all $(t, G, v) \in \mathcal{D}$ it holds

$$\text{dyn}(t, G, v) = \kappa\left(\left(\mathbf{T}_v(i)\right)_{i \in [t]}\right).$$

Proof. We show the proposition by proving both directions of the equivalence relation:

⇒: If there exists κ such that $\text{dyn}(t, G, v) = \kappa\left(\left(\mathbf{T}_v(i)\right)_{i \in [t]}\right)$ for all triplets $(t, G, v) \in \mathcal{D}$, then for any pair of nodes $u \in G_1, v \in G_2$ with $u \sim_{DUT} v$ it holds

$$\text{dyn}(t, G_1, u) = \kappa\left(\left(\mathbf{T}_u(i)\right)_{i \in [t]}\right) = \kappa\left(\left(\mathbf{T}_v(i)\right)_{i \in [t]}\right) = \text{dyn}(t, G_2, v).$$

⇐: On the other hand, if dyn preserves the unfolding equivalence, then we can define κ as

$$\kappa\left(\left(\mathbf{T}_v(i)\right)_{i \in [t]}\right) = \text{dyn}(t, G, v).$$

Note that the above equality is a correct specification for a function. In fact, if

$$\kappa\left(\left(\mathbf{T}_v(i)\right)_{i \in [t]}\right) = \kappa\left(\left(\mathbf{T}_u(i)\right)_{i \in [t]}\right)$$

implies $\text{dyn}(t, G, u) = \text{dyn}(t, G, v)$, then κ is uniquely defined.

□

B.2.4 Proof of Theorem 3.5

Since the proof proceeds analogously to the one in [88], we will only sketch the proof idea here and refer to the original paper for further details. The key point is that Theorem 3.5 is equivalent to another theorem ([93, Lemma 2]), where the domain is a finite subset of \mathcal{G} and the attributes are integers. Hence, proving this theorem is sufficient to prove Theorem 3.5.

To prove the equivalence between the two theorems, we need [93, Lemma 2], Theorem 3.2, and the extended version of [93, Lemma 1] to the domain of SAUHG \mathcal{G} , that we provide in the following. Intuitively, the lemma suggests that a domain of SAUHG with continuous attributes can be partitioned into small subsets so that the attributes of the graphs are almost constant in each partition. Moreover, in probability, a finite number of partitions is sufficient to cover a large portion of the domain.

Lemma B.1. *For any probability measure P on \mathcal{G} , and any reals λ, δ , where $\lambda > 0, \delta \geq 0$, there exists a real $\bar{b} > 0$, which is independent of δ , a set $\bar{\mathcal{G}} \subseteq \mathcal{G}$, and a finite number of partitions $\bar{\mathcal{G}}_1, \dots, \bar{\mathcal{G}}_p$ of $\bar{\mathcal{G}}$, where $\bar{\mathcal{G}}_j = \mathcal{G}_j \times \{v_j\}$, with $\mathcal{G}_j \subseteq \mathcal{G}$ and $v_j \in \mathcal{G}_j$, such that:*

- $P(\bar{\mathcal{G}}) \geq 1 - \lambda$ holds;
- for each j , all the graphs in \mathcal{G}_j have the same structure, i.e., they differ only in the values of their attributes;
- for each set $\bar{\mathcal{G}}_j$, there exists a hypercube $\mathcal{H}_j \subset \mathbb{R}^{NM2k}$ such that $\gamma_G \in \mathcal{H}_j$ holds for any graph $G' \in \mathcal{G}_j$ with $N = \max_{G' \in \mathcal{G}} |\mathcal{V}'|$ and $M = \max_{G' \in \mathcal{G}} |\mathcal{E}'|$. Here, $\gamma_{G'}$ denotes the vector obtained by concatenating all the attribute vectors of both nodes and edges of G' , namely $\gamma_{G'} = [A_{G'} | \Omega_{G'}]$, where $A_{G'}$ is the concatenation of all the node attributes and $\Omega_{G'}$ is the concatenation of all edge attributes;
- for any two different sets $\mathcal{G}_i, \mathcal{G}_j, i \neq j$, their graphs have different structures, or their hypercubes $\mathcal{H}_i, \mathcal{H}_j$ are disjoint, i.e., $\mathcal{H}_i \cap \mathcal{H}_j = \emptyset$;
- for each j and each pair of graphs $G_1, G_2 \in \mathcal{G}_j$, the inequality $\|\gamma_{G_1} - \gamma_{G_2}\|_\infty \leq \delta$ holds;
- for each graph $G' \in \bar{\mathcal{G}}$, the inequality $\|\gamma_{G'}\|_\infty \leq \bar{b}$ holds.

Proof. The proof is similar to the one contained in [93]. The only remark needed here is that we can consider the whole concatenating of all attributes from both nodes and edges without loss of generality; indeed, if we were considering the node and the edge attributes separately, we would need conditions on the hypercubes, s.t.:

$$\begin{aligned} \|A_{G_1} - A_{G_2}\|_\infty &\leq \delta^A, \delta^A > 0, \\ \text{and } \|\Omega_{G_1} - \Omega_{G_2}\|_\infty &\leq \delta^\Omega, \delta^\Omega > 0. \end{aligned}$$

Then we can stack those attribute vectors, as in the statement, s.t. :

$$\begin{aligned} \|\gamma_{G_1} - \gamma_{G_2}\|_\infty &= \|([A_{G_1} | \mathcal{F}] + [\mathcal{F} | \Omega_{G_1}]) - ([A_{G_2} | \mathcal{F}] + [\mathcal{F} | \Omega_{G_2}])\|_\infty \\ &\leq \|A_{G_1} - A_{G_2}\|_\infty + \|\Omega_{G_1} - \Omega_{G_2}\|_\infty \\ &\leq \delta^A + \delta^\Omega := \delta \end{aligned}$$

which allows us to exploit the same proof contained in [93]. □

Thanks to the previous lemma, adopting an argument similar to that in [93], it is proven that the following theorem, where the domain contains a finite number of graphs and the attributes are integers, is equivalent to Thm. 3.5.

Theorem B.2. For any finite set of p patterns $\{(G'_j, v) \mid G'_j \in \mathcal{G}, v \in \mathcal{V}'_j, j \in [p]\}$, with the maximal number of nodes in the domain $N = \max_{G' \in \mathcal{G}} |G'|$, for any function which preserves the attributed unfolding equivalence, and for any real $\varepsilon > 0$, there exist continuously differentiable functions $\text{AGGREGATE}^{(i)}$, $\text{COMBINE}^{(i)}$, $\forall i \leq 2N - 1$, s.t.

$$\mathbf{h}_v^i = \text{COMBINE}^{(i)} \left(\mathbf{h}_v^{(i-1)}, \text{AGGREGATE}^{(i)} \left(\{\mathbf{h}_u^{i-1}\}_{u \in ne_v}, \{\omega_{(u,v)}\}_{u \in ne_v} \right) \right)$$

and a function READOUT , with hidden dimension $r = 1$, i.e. $\mathbf{h}_v^i \in \mathbb{R}$, so that the function φ (realized by the AGNN), computed after N steps, satisfies the condition

$$|\tau(G'_j, v) - \varphi(G'_j, v)| \leq \varepsilon \quad \text{for any } v \in \mathcal{V}'_j. \quad (\text{B.5})$$

Sketch of the proof. The idea of the proof is designing a GNN that can approximate any function τ that preserves the attributed unfolding equivalence. According to Thm. 3.4 there exists a function κ , s.t.

$$\tau(G'_j, v) = \kappa(T_v).$$

Therefore, the GNN has to encode the attributed unfolding tree into the node attributes, i.e., for each node v , we want to have $\mathbf{h}_v = \nabla(\mathbf{T}_v)$, where ∇ is an encoding function that maps attributed unfolding trees into real numbers. The existence and injectiveness of ∇ are ensured by construction. More precisely, the encodings are constructed recursively by the $\text{AGGREGATE}^{(i)}$ and the $\text{COMBINE}^{(i)}$ functions using the neighborhood information, i.e., the node and edge attributes.

Consequently, the theorem can be proven given that there exist appropriate functions ∇ , $\text{AGGREGATE}^{(i)}$, $\text{COMBINE}^{(i)}$ and READOUT . For this purpose, the functions $\text{AGGREGATE}^{(i)}$ and $\text{COMBINE}^{(i)}$ must satisfy $\forall i \leq 2N - 1$:

$$\begin{aligned} \nabla(\mathbf{T}_v^i) &= \mathbf{h}_v^i \\ &= \text{COMBINE}^{(i)} \left(\mathbf{h}_v^{(i-1)}, \text{AGGREGATE}^{(i)} \left(\{\{\mathbf{h}_u^{i-1}\}\}_{u \in \mathcal{N}(v)}, \{\{\omega_{(u,v)}\}\}_{u \in \mathcal{N}(v)} \right) \right) \\ &= \text{COMBINE}^{(i)} \left(\nabla(\mathbf{T}_v^{i-1}), \text{AGGREGATE}^{(i)} \left(\{\{\nabla(\mathbf{T}_u^{i-1})\}\}_{u \in \mathcal{N}(v)}, \{\{\omega_{(u,v)}\}\}_{u \in \mathcal{N}(v)} \right) \right). \end{aligned}$$

In a simple solution, $\text{AGGREGATE}^{(i)}$ decodes the attributed trees of the neighbors u of v , \mathbf{T}_u^{i-1} , and stores them into a data structure to be accessed by $\text{COMBINE}^{(i)}$. The detailed construction of the appropriate functions is given in [88]. \square

B.2.5 Proof of Theorem 3.8

The proof works analogously to the one of Theorem 3.5. We just need some additional observations.

Lemma B.3. Lemma B.1 holds for the domain of dynamic graphs \mathcal{D} .

Proof. Indeed, taking into account the argument in [104], one can establish a bijection between the domain of dynamic graphs and the domain of SAUHGs; on the latter, we can directly apply B.1. \square

Again, Thm. 3.8 is equivalent to the following, where the domain contains a finite number of elements in \mathcal{D} and the attributes are integers. This time, we present the detailed construction because it is slightly different with respect to the one presented in [88].

Theorem B.4. For any finite set of p patterns $\{(t^{(j)}, G^{(j)}, v^{(j)}) \mid (t^{(j)}, G^{(j)}, v^{(j)}) \in \mathcal{D}, j \in [p]\}$ with the maximal number of nodes $N = \max_{G \in \mathcal{D}} |G|$ and with graphs having integer features, for any

measurable dynamical system preserving the unfolding equivalence, $\|\cdot\|$ be a norm on \mathbb{R}^m , P be any probability measure on \mathcal{D} and ϵ be any real number where $\epsilon > 0$. Then, there exists a DGNN as defined in Def. 3.4 such that the function φ (realized by this model) computed after N steps satisfies the condition

$$\|\text{dyn}(t^{(j)}, G^{(j)}, v^{(j)}) - \varphi(t^{(j)}, G^{(j)}, v^{(j)})\| \leq \epsilon \quad (\text{B.6})$$

$\forall j \in [p]$ where $t^{(j)} \in I$.

Proof. The proof of this theorem involves assuming that the output dimension is $m = 1$, i.e., $\text{dyn}(t, G, v) \in \mathbb{R}$, but the result can be extended to the general case with $m \in \mathbb{N}$ by concatenating the corresponding results. As a result of Theorem 3.7, there exists a function κ , s.t. $\text{dyn}(t, G, v) = g(x_v(t)) = \kappa((\mathbf{T}_v(i))_{i \in [t]})$, where $(\mathbf{T}_v(i))_{i \in [t]}$ is a sequence of $t + 1$ attributed unfolding trees. Given N_t as the number of nodes of the graph G at timestep t , in order to store the graph information, an attributed unfolding tree of depth $2N_t - 1$ is required for each node, in such a way that κ can satisfy

$$\text{dyn}(t, G, v) = \kappa((\mathbf{T}_v(i))_{i \in [t]}) = \kappa((\mathbf{T}_v^{2N_t - 1}(i))_{i \in [t]}).$$

The required depth is a straight consequence of [88, Theorem 4.1.3]. Considering the finite domain \mathcal{G} with $N = \max_{G \in \mathcal{G}} |G|$, using a depth of $2N - 1$, we are sure that every unfolding tree contains all the necessary information. Thus, from now on, we will assume that the depth of the unfolding trees is $2N - 1$. To simplify the notation, we will continue to use the notation $[\mathbf{T}_v(t_i)]_{i=1, \dots, j}$.

The main idea behind the proof of Theorem B.4 is to design a DGNN that can encode the sequence of attributed unfolding trees $(\mathbf{T}_v(i))_{i \in [t]}$ into the node attributes at each timestep t , i.e., $\mathbf{q}_v(t) = \#_t((\mathbf{T}_v(i))_{i \in [t]})$. This is achieved by using a coding function that maps sequences of $t + 1$ attributed trees into real numbers. To implement the encoding that could fit the definition of the DGNN, two coding functions are needed: the ∇ function, which encodes the attributed unfolding trees, and the family of coding functions $\#_t$. The composition of these functions is used to define the node's attributes, and the DGNN can produce the desired output by using this encoded information as follows:

$$\begin{aligned} \mathbf{q}_v(0) &= \mathbf{h}_v(0) = \#_0(\nabla^{-1}(\mathbf{h}_v(0))) \\ \mathbf{q}_v(t) &= \#_t(\text{APPEND}_t(\#_{t-1}^{-1}(\mathbf{q}_v(t-1)), \nabla^{-1}(\mathbf{h}_v(t)))) \end{aligned} \quad (\text{B.7})$$

where $\mathbf{h}_v := \mathbf{h}_v^{(L)}$ is the final representation produced by L layers of GNN. The auxiliary function APPEND_t and the ∇ , $\#_t$ coding functions are defined in the following.

The APPEND_t Function

Let \mathcal{T}_v^d be the domain of the attributed unfolding trees with root v , up to a certain depth d . The function $\text{APPEND}_t : \{(\mathbf{T}_v^d(i))_{i \in [t-1]}\} \cup \emptyset \times \mathcal{T}_v^d \rightarrow \{(\mathbf{T}_v^d(i))_{i \in [t]}\}$ is defined as follows:

$$\begin{aligned} \text{APPEND}_0(\emptyset, \mathbf{T}_v^d(0)) &:= \mathbf{T}_v^d(0) \\ \text{APPEND}_t((\mathbf{T}_v^d(0), \dots, \mathbf{T}_v^d(t-1)), \mathbf{T}_v^d(t)) \\ &:= (\mathbf{T}_v^d(0), \dots, \mathbf{T}_v^d(t-1), \mathbf{T}_v^d(t)) \end{aligned}$$

Intuitively, this function appends the unfolding tree snapshot of the node v at time t to the sequence of the unfolding trees of that node at the previous $t - 1$ timesteps.

In the following, the coding functions are defined; their existence and injectiveness are provided by construction.

The ∇ Coding Function

Let $\nabla := \mu_{\nabla} \circ \nu_{\nabla}$ be a composition of any two injective functions μ_{∇} and ν_{∇} with the following properties:

- μ_{∇} is an injective function from the domain of static unfolding trees, calculated on the nodes in the graph G_t , to the Cartesian product $\mathbb{N} \times \mathbb{N}^P \times \mathbb{Z}^A$, where P is the maximum number of nodes a tree could have and A is the attributes dimension.
- Intuitively, in the Cartesian product, \mathbb{N} represents the tree structure, \mathbb{N}^P denotes the node numbering, while, for each node, an integer vector in \mathbb{Z}^A is used to encode the node attributes. Notice that μ_{∇} exists and is injective since the maximal information contained in an unfolding tree is given by the union of all its node attributes and all its structural information, which equals the dimension of the codomain of μ_{∇} .
- ν_{∇} is an injective function from $\mathbb{N}^{P+1} \times \mathbb{Z}^A$ to \mathbb{R} , whose existence is guaranteed by the cardinality theory.

Since μ_{∇} and ν_{∇} are injective, also the existence and the injectiveness of ∇ is ensured.

The $\#_t$ Coding Functions

Similarly to ∇ , the functions $\#_t := \mu_{\#_t} \circ \nu_{\#_t}$ are composed by two functions $\mu_{\#_t}$ and $\nu_{\#_t}$ with the following properties:

- $\mu_{\#_t}$ is an injective function from the domain of the dynamic unfolding trees $\mathcal{T}_t^d(v) := \{(\mathbf{T}_v^d(i))_{i \in [t]}\}$ to the Cartesian product $\mathbb{N}^t \times \mathbb{N}^{tP_t} \times \mathbb{Z}^{tA} = \mathbb{N}^{t(P_t+1)} \times \mathbb{Z}^{tA}$, where P_t is the maximum number of nodes a tree could have at time t .
- $\nu_{\#_t}$ is an injective function from $\mathbb{N}^{t(P_t+1)} \times \mathbb{Z}^{tA}$ to \mathbb{R} , whose existence is guaranteed by the cardinality theory.

Since $\mu_{\#_t}$ and $\nu_{\#_t}$ are injective, also the existence and the injectiveness of $\#_t$ are ensured.

The recursive function f , and the functions $\text{AGGREGATE}_t^{(i)}$ and $\text{COMBINE}_t^{(i)}$

The recursive function f has to satisfy

$$f(\mathbf{q}_v(t-1), \mathbf{h}_v(t)) = \#_t((\mathbf{T}_v(i))_{i \in [t]}) = \mathbf{q}_v(t),$$

where the $\mathbf{h}_v(t)$ is the hidden representation of node v at time t extracted from the final layer of the t -th AGNN, i.e. $\mathbf{h}_v(t) = \text{AGNN}(G_t, v)$. In particular, at each iteration i , we have

$$\mathbf{h}_v^{(i)}(t) = \text{COMBINE}_t^{(i)}\left(\mathbf{h}_v^{(i-1)}(t), \text{AGGREGATE}_t^{(i)}\left(\{\{\mathbf{h}_u^{(i-1)}(t)\}_{u \in \mathcal{N}_v(t)}, \{\{\omega_{(u,v)}(t)\}_{u \in \mathcal{N}_v(t)}\}\right)\right)$$

Further, the functions $\text{AGGREGATE}_t^{(i)}$ and $\text{COMBINE}_t^{(i)}$ – following the proof in [88] – must satisfy

$$\begin{aligned} \nabla(\mathbf{T}_v^i(t)) &= \mathbf{h}_v^i(t) = \text{COMBINE}_t^{(i)}\left(\mathbf{h}_v^{i-1}(t), \text{AGGREGATE}_t^{(i)}\left(\{\{\mathbf{h}_u^{i-1}(t)\}_{u \in \mathcal{N}_v(t)}, \{\{\omega_{(u,v)}(t)\}_{u \in \mathcal{N}_v(t)}\}\right)\right) \\ &= \text{COMBINE}_t^{(i)}\left(\nabla(\mathbf{T}_v^{i-1}(t)), \text{AGGREGATE}_t^{(i)}\left(\{\{\nabla(\mathbf{T}_u^{i-1}(t))\}_{u \in \mathcal{N}_v(t)}\}\right)\right) \end{aligned}$$

$\forall i \leq 2N - 1$ and $\forall t \in I$.

For example, the trees can be collected into the coding of a new tree, i.e.,

$$\text{AGGREGATE}_t^{(i)}(\nabla(\mathbf{T}_u^{i-1}(t)), u \in \mathcal{N}_v(t)) = \nabla(\cup_{u \in \mathcal{N}_v(t)} \nabla^{-1}(\nabla(\mathbf{T}_u^{i-1}(t))))$$

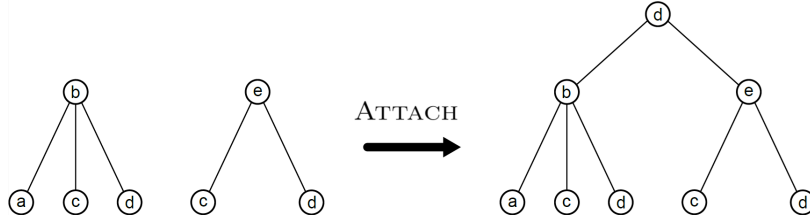


Figure B.1: The ATTACH operator on trees.

where $\cup_{u \in \mathcal{N}_v(t)}$ denotes an operator that constructs a tree with a root having void attributes from a set of subtrees (see Fig. B.1). Then, $\text{COMBINE}_t^{(i)}$ assigns the correct attributes to the root by extracting them from $\mathbf{T}_v^{i-1}(t)$, i.e.,

$$\text{COMBINE}_t^{(i)}(\nabla(\mathbf{T}_v^{i-1}(t)), b) = \nabla(\text{ATTACH}(\nabla^{-1}(\nabla(\mathbf{T}_v^{i-1}(t))), \nabla^{-1}(b))),$$

where ATTACH is an operator that returns a tree constructed by replacing the attributes of the root in the latter tree with those of the former tree and b is the result of the $\text{AGGREGATE}_t^{(i)}$ function. Now, notice that, with this definition, $\text{AGGREGATE}_t^{(i)}$, $\text{COMBINE}_t^{(i)}$, and $\text{READOUT}_{\text{dyn}}$ may not be differentiable. Nevertheless, Eq. (B.6) has to be satisfied only for a finite number of graphs, namely G_j . Thus, we can specify other functions $\overline{\text{AGGREGATE}}_t^{(i)}$, $\overline{\text{COMBINE}}_t^{(i)}$, and $\overline{\text{READOUT}}$, which produce exactly the same computations when they are applied on the graphs G_j , but that can be extended to the rest of their domain so that they are continuously differentiable. Obviously, such an extension exists since those functions are only constrained to interpolate a finite number of points³.

The $\text{READOUT}_{\text{dyn}}$ function

Eventually, $\text{READOUT}_{\text{dyn}}$ must satisfy:

$$\kappa(\cdot) := \text{READOUT}_{\text{dyn}}(\#_t(\cdot))$$

so that, ultimately,

$$\begin{aligned} \text{dyn}(t, G, v) = \\ \text{READOUT}_{\text{dyn}}(\#_t(\text{APPEND}_t(\#_{t-1}^{-1}(\mathbf{q}_v(t-1)), \nabla^{-1}(\mathbf{h}_v(t)))))) \end{aligned}$$

□

This concludes the proof of Theorem B.4 and because of the equivalence between the two results, it also concludes the proof for 3.8.

B.2.6 Proof of Theorem 3.9

Statement. Assume that the hypotheses of Thm. 3.8 are fulfilled and $\mathcal{Q}_{\mathcal{D}}$ is a class of discrete DGNNs with universal components. Then, there exists a parameter set θ , and the functions

³Notice that a similar extension can also be applied to the coding function ∇ and to the decoding function ∇^{-1} . In this case, the coding function is not injective on the whole domain, but only on the graphs mentioned in the theorem.

$\text{AGNN}(t)_\theta, f_{0,\theta}, f_\theta$, implemented by Neural Networks in $\mathcal{Q}_{\mathcal{D}}$, such that the thesis of Thm. 3.8 holds.

Proof. The idea of the proof follows from the same reasoning adopted in [88]. Intuitively, since the discrete DGNN of Thm. 3.8 is implemented by continuously differentiable functions, its output depends continuously on the possible changes in the DGNN implementation: small changes in the function implementation cause small changes in the DGNN outputs. Therefore, the functions of the DGNN of Thm. 3.8 can be replaced by Neural Networks, provided that those networks are suitable approximators.

As in the proof of the dynamic version of the approximation theorem, cf. Thm. 3.8, without loss of generality, we will assume that the attribute dimension is $n = 1^4$.

First of all, note that Thm. 3.8 ensures that we can find continuously differentiable functions $\bar{f}, \overline{\text{READOUT}}_{\text{dyn}}$ such that, for the corresponding function $\bar{\varphi}$ implemented by the DGNN it holds:

$$P(\|\text{dyn}(t, G, v) - \bar{\varphi}(t, G, v)\| \leq \frac{\epsilon}{2}) \geq 1 - \lambda \quad \forall t \in I, \epsilon, \lambda > 0. \quad (\text{B.8})$$

Considering that the theorem has to hold only in probability, we can also assume that the domain is bounded to a finite set of patterns $\{(t^{(i)}, (G_t)_{t \in I}^{(i)}, v^{(i)}) \mid i = 1, \dots, p\}$ (as in Theorem B.4). As a result, the functions \bar{f} and $\overline{\text{READOUT}}_{\text{dyn}}$ are bounded and have a bounded Jacobian. We can take the maximum of these Jacobians, which we will denote as B .

Moreover, let $f_\theta, \overline{\text{READOUT}}_{\text{dyn}, \theta}$ be universal components for DGNN, as in Def. 3.17, that approximate $\bar{f}, \overline{\text{READOUT}}_{\text{dyn}}$, respectively. Further, let $\epsilon_1, \epsilon_2, > 0$ be the corresponding approximation errors, i.e.,

$$\begin{aligned} \|\bar{f}(\mathbf{q}, \mathbf{h}) - f_\theta(\mathbf{q}, \mathbf{h})\|_\infty &\leq \epsilon_1, \text{ and} \\ \|\overline{\text{READOUT}}_{\text{dyn}}(Q(t)) - \overline{\text{READOUT}}_{\text{dyn}, \theta}(Q(t))\|_\infty &\leq \epsilon_2 \end{aligned} \quad (\text{B.9})$$

hold $\forall t \in I$.

Now, from the proof of Theorem 3.6 we know that

$$P(\|\overline{\text{AGNN}}_i(G, v) - \text{AGNN}_{\theta, i}(G, v)\| \leq \epsilon_s) \geq 1 - \lambda_i$$

for $i \in [t]$, $\epsilon_s > 0$ and for any norm. Then we can take every λ_i small enough, s.t.

$$\|\overline{\text{AGNN}}_i(G, v) - \text{AGNN}_{\theta, i}(G, v)\|_\infty \leq \epsilon_s$$

holds on a finite set of patterns large enough to include those ones of the i -th timestep of each patterns of dynamic graphs on which Eq. (B.8) holds.

Therefore, if we define $\bar{\mathbf{h}}(t) := \overline{\text{AGNN}}_i(G_t)$ and $\mathbf{h}_\theta(t) := \text{AGNN}_{\theta, i}(G_t)$ we have

$$\|\bar{\mathbf{h}}(t) - \mathbf{h}_\theta(t)\|_\infty = \|\overline{\text{AGNN}}_i(G_t) - \text{AGNN}_{\theta, i}(G_t)\|_\infty \leq \epsilon_s.$$

In addition, let $\bar{H}(t)$ and $H_\theta(t)$ be the internal representations produced by $\overline{\text{AGNN}}$ and AGNN_θ , stacked over all the nodes of the input graph. Then it holds

$$\|\bar{H}(t) - H_\theta(t)\|_\infty \leq N\epsilon_s \quad \forall t \in I, \quad (\text{B.10})$$

where $N = \max_{G \in \mathcal{D}} |G|$ is the maximum number of nodes of the static graphs input in the bounded domain. Let again $\bar{Q}(0) := \bar{H}(0)$ and $\bar{Q}(t) := \bar{F}(\bar{Q}(t-1), \bar{H}(t))$ be the stacking of the internal states produced by DGNN's internal recursive function \bar{f} . Analogously, let $Q_\theta(0) :=$

⁴A GNN can theoretically be modeled with multiple components by stacking Neural Networks for each dimension, respectively.

$H_\theta(0)$ and $Q_\theta(t) := F_\theta(Q_\theta(t-1), H_\theta(t))$ be the output produced by the corresponding function of the parameterized DGNN.

Then it holds:

$$\|\bar{Q}(0) - Q_\theta(0)\|_\infty = \|\bar{H}(0) - H_\theta(0)\|_\infty \leq N\epsilon_s \quad (\text{B.11})$$

and

$$\begin{aligned} \|\bar{f}(\bar{Q}(0), \cdot) - \bar{f}(Q_\theta(0), \cdot)\|_\infty &\leq B\|\bar{Q}(0) - Q_\theta(0)\|_\infty \\ \|\bar{f}(\cdot, \bar{H}(1)) - \bar{f}(\cdot, H_\theta(1))\|_\infty &\leq B\|\bar{H}(1) - H_\theta(1)\|_\infty \end{aligned}$$

for a bound B on the Jacobian of $\bar{f}(\mathbf{q}, \mathbf{h}) \forall t \in I$ and $\forall \mathbf{q}$, which, along with Eq. (B.10) and Eq. (B.11) gives

$$\begin{aligned} \|\bar{f}(\bar{Q}(0), \cdot) - \bar{f}(Q_\theta(0), \cdot)\|_\infty &\leq N\epsilon_s B \\ \|\bar{f}(\cdot, \bar{H}(1)) - \bar{f}(\cdot, H_\theta(1))\|_\infty &\leq N\epsilon_s B \end{aligned} \quad (\text{B.12})$$

Therefore, we have that:

$t = 1$:

$$\begin{aligned} &\|\bar{Q}(1) - Q_\theta(1)\|_\infty \\ &= \|\bar{f}(\bar{Q}(0), \bar{H}(1)) - f_\theta(Q_\theta(0), H_\theta(1))\|_\infty \\ &\stackrel{\text{add } 0}{=} \|\bar{f}(\bar{Q}(0), \bar{H}(1)) - \bar{f}(Q_\theta(0), \bar{H}(1)) \\ &\quad + \bar{f}(Q_\theta(0), \bar{H}(1)) - \bar{f}(Q_\theta(0), H_\theta(1)) \\ &\quad + \bar{f}(Q_\theta(0), H_\theta(1)) - f_\theta(Q_\theta(0), H_\theta(1))\|_\infty \\ &\stackrel{\Delta\text{-ineq.}}{\leq} \|\bar{f}(\bar{Q}(0), \bar{H}(1)) - \bar{f}(Q_\theta(0), \bar{H}(1))\|_\infty \\ &\quad + \|\bar{f}(Q_\theta(0), \bar{H}(1)) - \bar{f}(Q_\theta(0), H_\theta(1))\|_\infty \\ &\quad + \|\bar{f}(Q_\theta(0), H_\theta(1)) - f_\theta(Q_\theta(0), H_\theta(1))\|_\infty \\ &\stackrel{(\text{B.12})}{\leq} 2N\epsilon_s B + N\epsilon_1 \\ &:= \lambda_1(\epsilon_s, \epsilon_1). \end{aligned}$$

$t > 0$: Analogously, it follows for $t \geq 1$ that

$$\begin{aligned} &\|\bar{Q}(t) - Q_\theta(t)\|_\infty \\ &= \|\bar{f}(\bar{Q}(t-1), \bar{H}(t)) - f_\theta(Q_\theta(t-1), H_\theta(t))\|_\infty \\ &= \|\bar{f}(\bar{Q}(t-1), \bar{H}(t)) - \bar{f}(Q_\theta(t-1), \bar{H}(t)) \\ &\quad + \bar{f}(Q_\theta(t-1), \bar{H}(t)) - \bar{f}(Q_\theta(t-1), H_\theta(t)) \\ &\quad + \bar{f}(Q_\theta(t-1), H_\theta(t)) - f_\theta(Q_\theta(t-1), H_\theta(t))\|_\infty \\ &\leq \|\bar{f}(\bar{Q}(t-1), \bar{H}(t)) - \bar{f}(Q_\theta(t-1), \bar{H}(t))\|_\infty \\ &\quad + \|\bar{f}(Q_\theta(t-1), \bar{H}(t)) - \bar{f}(Q_\theta(t-1), H_\theta(t))\|_\infty \\ &\quad + \|\bar{f}(Q_\theta(t-1), H_\theta(t)) - f_\theta(Q_\theta(t-1), H_\theta(t))\|_\infty \\ &\leq N\lambda_0 B + N\epsilon_s B + N\epsilon_1 \\ &:= \lambda_1(\epsilon_s, \epsilon_1). \end{aligned}$$

The above reasoning can then be applied recursively to prove that

$$\|\bar{Q}(t) - Q_\theta(t)\|_\infty \leq \lambda_t(\epsilon_s, \epsilon_1),$$

where $\lambda_t(\epsilon_s, \epsilon_1)$ could be found as little as possible, according to ϵ_s, ϵ_1 . Finally, let $\epsilon_2 > 0$, so that

$$\begin{aligned} & \|\bar{\varphi}(t, G, v) - \varphi_\theta(t, G, v)\|_\infty \\ &= \|\overline{\text{READOUT}}_{\text{dyn}}(\bar{Q}(t)) - \text{READOUT}_{\text{dyn}, \theta}(\mathbf{Q}_\theta(t))\|_\infty \\ &\leq \|\overline{\text{READOUT}}_{\text{dyn}}(\bar{Q}(t)) - \overline{\text{READOUT}}_{\text{dyn}}(\mathbf{Q}_\theta(t))\|_\infty \\ &\quad + \|\overline{\text{READOUT}}_{\text{dyn}}(\mathbf{Q}_\theta(t)) - \text{READOUT}_{\text{dyn}, \theta}(\mathbf{Q}_\theta(t))\|_\infty \\ &\leq \lambda_t B + \epsilon_2 = \lambda(\epsilon_s, \epsilon_1, \epsilon_2). \end{aligned}$$

Thus, we choose $\epsilon_s, \epsilon_1, \epsilon_2$, s.t. $\lambda \leq \frac{\epsilon}{2}$; going back in probability, we obtain

$$P(\|\bar{\varphi}(t, G, v) - \varphi_\theta(t, G, v)\| \leq \frac{\epsilon}{2}) \geq 1 - \lambda \quad \forall t \in I,$$

which, along with Eq. (B.8), proves the result. \square

B.3 WL Hierarchy

Here is reported the proof of Theorem 4.8 from Chapter 4.

B.3.1 Proof of Theorem 4.8

Statement. 1-DWL \equiv 1-AWL

Proof. Let $G = (G_t)_{t \in I}$ be a dynamic graph and $\text{STATIC}(G) =: G' = (\mathcal{V}', \mathcal{E}', \alpha', \omega')$ the SAUHG resulting from a bijective graph type transformation (cf. Appendix B.1). Furthermore, let $\mathcal{V} := \bigcup_{t \in I} \mathcal{V}_t$ be the set of all nodes appearing in the graph sequence of G and $\tilde{\alpha} : \mathcal{V} \times I \rightarrow \mathcal{A} \times \perp$ with $\tilde{\alpha}_v(t) := (\alpha_v(t), \rho)$ be the extended attribute function for all nodes including a flag $\rho \in \{0, 1\}$ for the existence of a node at time t . The theorem follows immediately from

$$\left(\mathbf{c}_u^{(i)}(t) \right)_{t \in I} = \left(\mathbf{c}_v^{(i)}(t) \right)_{t \in I} \Leftrightarrow \mathbf{c}_u^{(i)} = \mathbf{c}_v^{(i)},$$

for all iterations i and $u, v \in \mathcal{V}$. By induction, it follows:

$i = 0$:

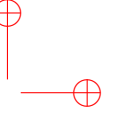
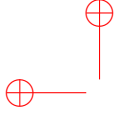
$$\begin{aligned} & \left(\mathbf{c}_u^{(0)}(t) \right)_{t \in I} = \left(\mathbf{c}_v^{(0)}(t) \right)_{t \in I} \stackrel{\text{Def. 3.9}}{\Leftrightarrow} \left(\text{Hash}(\tilde{\alpha}_u(t)) \right)_{t \in I} = \left(\text{Hash}(\tilde{\alpha}_v(t)) \right)_{t \in I} \\ & \Leftrightarrow \tilde{\alpha}_u(t) = \tilde{\alpha}_v(t) \quad \forall t \in I \stackrel{\text{by constr.}}{\Leftrightarrow} \alpha'_u = \alpha'_v \Leftrightarrow \text{Hash}(\alpha'_u) = \text{Hash}(\alpha'_v) \\ & \Leftrightarrow \mathbf{c}_u^{(0)} = \mathbf{c}_v^{(0)} \end{aligned}$$

$i > 0$: Assume the induction hypothesis (IH) is true for $i - 1$ and show the assumption is also

true for i .

$$\begin{aligned}
& \left(\mathbf{c}_u^{(i)}(t) \right)_{t \in I} = \left(\mathbf{c}_v^{(i)}(t) \right)_{t \in I} \stackrel{\text{Def. 3.9}}{\Leftrightarrow} \left(\text{Hash} \left(\mathbf{c}_u^{(i-1)}(t), \{\!\{ \mathbf{c}_n^{(i-1)}(t) \}\!\}_{n \in \mathcal{N}_u(t)}, \{\!\{ \omega_{(u,n)}(t) \}\!\}_{n \in \mathcal{N}_u(t)} \right) \right)_{t \in I} \\
& = \left(\text{Hash} \left(\mathbf{c}_v^{(i-1)}(t), \{\!\{ \mathbf{c}_{n'}^{(i-1)}(t) \}\!\}_{n' \in \mathcal{N}_v(t)}, \{\!\{ \omega_{(v,n')} (t) \}\!\}_{n' \in \mathcal{N}_v(t)} \right) \right)_{t \in I} \\
& \stackrel{\text{Bij. of Hash}}{\Leftrightarrow} \mathbf{c}_u^{(i-1)}(t) = \mathbf{c}_v^{(i-1)}(t) \wedge \{\!\{ \mathbf{c}_n^{(i-1)}(t) \}\!\}_{n \in \mathcal{N}_u(t)} = \{\!\{ \mathbf{c}_{n'}^{(i-1)}(t) \}\!\}_{n' \in \mathcal{N}_v(t)} \wedge \\
& \quad \wedge \{\!\{ \omega_{(u,n)}(t) \}\!\}_{n \in \mathcal{N}_u(t)} = \{\!\{ \omega_{(v,n')} (t) \}\!\}_{n' \in \mathcal{N}_v(t)} \quad \forall t \in I \\
& \stackrel{\text{IH}}{\Leftrightarrow} \mathbf{c}_u^{(i-1)} = \mathbf{c}_v^{(i-1)} \wedge \{\!\{ \mathbf{c}_n^{(i-1)} \}\!\}_{n \in \mathcal{N}_u} = \{\!\{ \mathbf{c}_{n'}^{(i-1)} \}\!\}_{n' \in \mathcal{N}_v} \wedge \{\!\{ \omega_{(u,n)} \}\!\}_{n \in \mathcal{N}_u} = \{\!\{ \omega_{(v,n')} \}\!\}_{n' \in \mathcal{N}_v} \\
& \stackrel{\text{Bij. of Hash}}{\Leftrightarrow} \text{Hash} \left(\left(\mathbf{c}_u^{(i-1)}, \{\!\{ \mathbf{c}_n^{(i-1)} \}\!\}_{n \in \mathcal{N}_u}, \{\!\{ \omega_{(u,n)} \}\!\}_{n \in \mathcal{N}_u} \right) \right) = \text{Hash} \left(\left(\mathbf{c}_v^{(i-1)}, \{\!\{ \mathbf{c}_{n'}^{(i-1)} \}\!\}_{n' \in \mathcal{N}_v}, \{\!\{ \omega_{(v,n')} \}\!\}_{n' \in \mathcal{N}_v} \right) \right) \\
& \stackrel{\text{Def. 3.5}}{\Leftrightarrow} \mathbf{c}_u^{(i)} = \mathbf{c}_v^{(i)}
\end{aligned}$$

□



Appendix C

Appendix

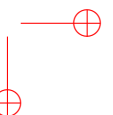
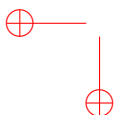
This appendix contains the proofs of all the theorems sketched in Chapter 5.

C.1 Additional Related work

In this section, we provide a more comprehensive discussion on related work.

Walks vs. Paths. Although walks and paths in a graph are rather similar concepts, they have different implications for GNN expressivity. By definition, walks are sequences of adjacent nodes, and therefore paths are particular walks without node repetitions. However, due to the local nature of the message-passing procedure, which iteratively updates the feature of a node based on its neighborhood, GNNs at layer ℓ encode all the walks up to length ℓ for each node (see [89]). Conversely, GNNs do not encode the identity of a node and therefore cannot identify repetitions of the same node in a walk. We thus argue that aggregating paths, in contrast to walks, does contribute additional information that can be helpful to distinguish between non-isomorphic graphs. Other approaches utilize walks in a way that increases the expressive power of GNNs. For example, [182] proposes the walk Message Passing Neural Network (walk-MPNN), generalizing the second-order non-linear invariant network by [183]. This model computes embeddings for pairs of nodes by encoding the walks between the two nodes. This architecture is strictly more powerful than 1-WL but is bounded in expressive power by 3-WL, regardless of the length of the walks. Indeed, increasing the length may allow to distinguish graphs with less iterations but cannot enhance expressivity. Conversely, the power of PATH-WL increases with the length of the paths. Therefore, PAIN is more powerful than walk-MPNN for some graph families such as strongly regular graphs (see Theorem 5.4).

Path-based GNNs. As argued in the previous paragraph, using paths to update node embeddings in GNNs can increase their expressive power. Several architectures have been devised that encode path information in different ways. For instance, [121] propose shortest path networks, which replace the topological neighborhood with shortest paths. They are provably more expressive than 1-WL, but not more expressive than 3-WL. [122] consider geodesic information between pairs of nodes and show that they distinguish almost all d -regular graphs for which 1-WL consistently fails. Closely related to using shortest paths directly is the incorporation of distance information. This can be done either explicitly ([184, 123, 124]) or implicitly ([125, 126, 127]). [123, 184, 124] propose position aware GNNs, whereas [125] introduce a transformer architecture with spatial encoding. These approaches are quite similar to the general idea of using positional encodings in the context of graph representation learning ([145, 185, 186]). Recently, several GNNs have been proposed that consider *all* paths instead ([128, 130, 129]). [129] introduce a color refinement test based on path complexes, a topological generalization of paths, which is strictly more expressive than 1-WL and not bounded by 3-WL. [128] propose a graph neural



network which samples from the set of all paths. Their model additionally learns structure and distance information with the help of a recurrent cell. [130] propose pathNN, which aggregates paths instead of the standard topological neighborhood. They achieve strong empirical performance by additionally encoding shortest path distances in an LSTM cell. Please refer to the discussion of Remark C.1 for a detailed comparison between PATH-WL and pathNN.

k -Hop GNNs. Standard message-passing GNNs aggregate messages from the direct neighbors of each node, which are called the *first-hop* neighbors. Recently, many architectures have generalized the message-passing scheme by aggregating information from all the nodes within the k -hop neighborhood simultaneously ([126, 187, 188, 189]). The idea of aggregating distant nodes, beyond the first-order neighborhood, is similar to that of path-based GNNs. However, in the k -hop aggregation we gather nodes with the same shortest path distance from a reference node, which is at most k . In contrast, the path-based aggregation provides a richer and context-aware representation, as it includes information such as the order of nodes within the same k -hop. [127] and [190] extended the k -hop GNNs by encoding, respectively, the subgraph induced by the nodes in the k -th hop (KP-GNN) and the subgraph induced by the whole k -hop neighborhood (N_k). All these modifications increase the expressive power of GNNs beyond 1-WL [127, Prop.1]. The k -hop GNNs are limited by 3-WL, including the KP-GNN [127, Thm. 2]. On the contrary, considering the complete k -hop increases the expressivity because N_k is incomparable to 3-WL [190, Thm. 6.3]. However, d -PATH-WL and N_k are fundamentally different graph coloring procedures: d -PATH-WL aggregates only paths up to a certain length which are endowed with distance information, while N_k aggregates the subgraph induced by the entire k -hop neighborhood.

C.2 Additional preliminaries

In the following, we provide some additional preliminaries.

Definition C.1 ([139]). Let $G = (\mathcal{V}_G, \mathcal{E}_G)$ and $H = (\mathcal{V}_H, \mathcal{E}_H)$ be two graphs. We say that H is a *homomorphic image* of G if there is a surjective homomorphism $\phi : \mathcal{V}_G \rightarrow \mathcal{V}_H$ such that:

$$\mathcal{E}_H = \{(\phi(u), \phi(v)) \mid (u, v) \in \mathcal{E}_G\}$$

We denote with $\text{spasm}(G)$ the set of homomorphic images of G .

The tree-width of a graph H , noted by $\text{tw}(H)$, can be defined in terms of tree decomposition.

Definition C.2. ([191]) A tree decomposition of a graph $G = (\mathcal{V}, \mathcal{E})$ is a pair (X, T) where $T = (I, A)$ is a tree, and $X = \{X_i : i \in I\}$ is a family of subsets of \mathcal{V} , such that:

- (i) $\bigcup_{i \in I} X_i = \mathcal{V}$
- (ii) Every edge of G has both its ends in some X_i , for $i \in I$.
- (iii) For all $i, j, k \in I$, if j lies on the path from i to k in T , then $X_i \cap X_k \subseteq X_j$.

The tree-width of a tree decomposition is $\max_{i \in I} |X_i| - 1$. The tree-width of G is the minimum tree-width taken over all possible tree decompositions of G .

Definition C.3 ([139]). The hereditary tree-width of G , noted by $\text{hdtw}(G)$, is the maximum tree-width among the tree-widths of the homomorphic images of G , i.e.,

$$\text{hdtw}(G) := \max_{H \in \text{spasm}(G)} \text{tw}(H)$$

Definition C.4 ([192]). An **Eulerian cycle**, also called an Eulerian circuit or Euler tour, in an undirected graph is a cycle that uses each edge exactly once. If such a cycle exists, the graph is called *Eulerian*.

C.3 Path-WL: A path-based WL test

C.3.1 Proof of Proposition 5.2

Statement. For every $d' \geq d \geq 0$, $\ell' \geq \ell \geq 1$ it holds that

$$d\text{-PATH-WL}^\ell \sqsubseteq d'\text{-PATH-WL}^\ell,$$

and

$$d\text{-PATH-WL}^\ell \sqsubseteq d\text{-PATH-WL}^{\ell'}.$$

Proof. Due to the transitivity of the order relation \sqsubseteq , it suffices to prove the statement for $d' = d + 1$ and $\ell' = \ell + 1$. To demonstrate the non-decreasing expressive power of $d\text{-PATH-WL}^\ell$ with respect to d , first we need to show that for every ℓ , if two nodes u and v are s.t. $u \sim_{(d+1)\text{-PATH-WL}^\ell} v$ then $u \sim_{d\text{-PATH-WL}^\ell} v$. Let $\mathbf{c}_{v_i}^d := \mathbf{Hash}(d\text{-}\mathcal{P}_{v_i}^\ell)$ be the color of node v_i after termination of $d\text{-PATH-WL}$. Hence, $\mathbf{c}_u^{d+1} = \mathbf{c}_v^{d+1}$ by hypothesis. Due to the injectivity of the **HASH** function, this implies that the multiset of paths with distance encoding up to length $d + 1$ are equal, that is $(d+1)\text{-}\mathcal{P}_u^\ell = (d+1)\text{-}\mathcal{P}_v^\ell$. What we aim to infer is that $d\text{-}\mathcal{P}_u^\ell = d\text{-}\mathcal{P}_v^\ell$. By definition, for every path $((\mathbf{c}_v, \eta_{vv}^d), (\mathbf{c}_{v_1}, \eta_{vv_1}^d), \dots, (\mathbf{c}_{v_\ell}, \eta_{vv_\ell}^d)) \in d\text{-}\mathcal{P}_v^\ell$, $\eta_{vv_i}^d \neq \emptyset$ if $\eta_{vv_i}^d \leq d < d + 1$. Therefore, it holds for any path in $(d+1)\text{-}\mathcal{P}_v^\ell$ that all tuples $(\mathbf{c}_{v_i}, \eta_{vv_i}^{d+1})$ are the same in $d\text{-}\mathcal{P}_v^\ell$ if the distance between the nodes v and v_i is less or equal than d , otherwise $\eta_{vv_i}^d$ is \emptyset . Hence, we conclude that $d\text{-}\mathcal{P}_u^\ell = d\text{-}\mathcal{P}_v^\ell$. To prove the monotonicity of $d\text{-PATH-WL}^\ell$ as the path length increases, it is enough to consider two facts. The first is the injectivity of the **Hash** function, and the second is that $d\text{-}\mathcal{P}_v^\ell$ contains paths *up to length* ℓ . Hence, if we can distinguish two nodes with a certain length ℓ , these different paths are also included in $d\text{-}\mathcal{P}_v^{\ell+1}$. \square

Remark C.1. For every $\ell \geq 1$, it holds that $0\text{-PATH-WL}^{\ell,(\ell)} \sqsupseteq \text{pathNN}$ ([130]) with path length ℓ .

Discussion of Remark C.1. We compare $d\text{-PATH-WL}$ and the annotation scheme characterizing pathNN with respect to the same number of iterations ℓ . We claim that one iteration of $d\text{-PATH-WL}$ contains all information of the corresponding layer of pathNN. For example, in the first layer pathNN computes paths of length one (= the neighborhood) whereas 0-PATH-WL computes paths of length up to ℓ . At the second layer, pathNN aggregates paths of length two where each node feature is updated given the paths of length one, computed at the first layer. 0-PATH-WL aggregates the same set of paths up to length ℓ , but each node is updated with the paths of length up to ℓ , computed in the first layer. After the second iteration of 0-PATH-WL , one node may receive information from nodes at distance 2ℓ from it. At each step, 0-PATH-WL has access to all information that pathNN contains, with the same asymptotic complexity. Hence, the expressivity of 0-PATH-WL cannot be bounded by pathNN. Moreover, in the pathNN framework, the length of the paths must coincide with the number of iterations while these two parameters are independent in our algorithm. We often observe that these parameters positively affect each other: increasing the length may decrease the number of iterations needed to distinguish two graphs, and vice versa (see Section C.7). \square

C.4 Relation to the k -WL Hierarchy.

C.4.1 Proof of Theorem 5.3

Statement. For every path length $\ell > 1$ and every $d \geq 0$, d -PATH-WL $^\ell$ is more expressive than 1-WL.

Proof. It is sufficient to prove the theorem for $d = 0$ because the addition of distance information $d > 0$ would not decrease the expressive power of the test. The proof consists of two parts: (i) we demonstrate that 0-PATH-WL is as expressive as 1-WL, and (ii) we prove that 0-PATH-WL is more expressive than 1-WL. For (i), we prove that for every length $\ell > 1$, for every node u, v and every iteration $i \in \mathbb{N}$,

$$u \approx_{WL^{(i)}} v \Rightarrow u \approx_{0\text{-PATH-WL}^{\ell,(i)}} v.$$

By the definition of the two coloring algorithms, this corresponds to

$$\text{Hash}\left(\mathbf{c}_v^{(i)}, \left\{ \left\{ \mathbf{c}_w^{(i)} \mid w \in N(v) \right\} \right\}\right) \neq \text{Hash}\left(\mathbf{c}_u^{(i)}, \left\{ \left\{ \mathbf{c}_w^{(i)} \mid w \in N(u) \right\} \right\}\right) \Rightarrow \text{Hash}(0\text{-}\mathcal{P}_v^{\ell,(i)}) \neq \text{Hash}(0\text{-}\mathcal{P}_u^{\ell,(i)})$$

Let α be the left-hand side of the implication. Due to the injectivity of the **Hash** function it is sufficient to show that α implies $0\text{-}\mathcal{P}_v^{\ell,(i)} \neq 0\text{-}\mathcal{P}_u^{\ell,(i)}$. α is true if $\mathbf{c}_v^{(i)} \neq \mathbf{c}_u^{(i)}$ or if $\left\{ \left\{ \mathbf{c}_w^{(i)} \mid w \in N(v) \right\} \right\} \neq \left\{ \left\{ \mathbf{c}_w^{(i)} \mid w \in N(u) \right\} \right\}$, or both. Given that every element of $0\text{-}\mathcal{P}_v^{\ell,(i)}$ is a sequence whose first element is $\mathbf{c}_v^{(i)}$, i.e.,

$$0\text{-}\mathcal{P}_v^{\ell,(i)} := \left\{ \left\{ (\mathbf{c}_v^{(i)}) \right\} \right\} \cup \left\{ \left\{ (\mathbf{c}_v^{(i)}, \mathbf{c}_w^{(i)})_{w \in N(v)} \right\} \right\} \cup \dots \cup \left\{ \left\{ (\mathbf{c}_v^{(i)}, \mathbf{c}_w^{(i)}, \dots, \mathbf{c}_y^{(i)})_{w \in N(v) \wedge y = \pi_\ell(p_v^\ell)} \right\} \right\},$$

where $\pi_j(p_v^\ell)$ denotes the j -th node of path p_v^ℓ , we can simply conclude that if $\mathbf{c}_v^{(i)} \neq \mathbf{c}_u^{(i)}$ then $0\text{-}\mathcal{P}_v^{\ell,(i)} \neq 0\text{-}\mathcal{P}_u^{\ell,(i)}$. Suppose that $\mathbf{c}_v^{(i)} = \mathbf{c}_u^{(i)}$. Then, $\left\{ \left\{ \mathbf{c}_w^{(i)} \mid w \in N(v) \right\} \right\} \neq \left\{ \left\{ \mathbf{c}_w^{(i)} \mid w \in N(u) \right\} \right\}$ implies $\left\{ \left\{ (\mathbf{c}_v^{(i)}, \mathbf{c}_w^{(i)})_{w \in N(v)} \right\} \right\} \neq \left\{ \left\{ (\mathbf{c}_u^{(i)}, \mathbf{c}_w^{(i)})_{w \in N(u)} \right\} \right\}$. Due to the fact that $0\text{-}\mathcal{P}^{\ell,(i)}$ is a multiset of sequences of heterogeneous length, the fact that paths of length one are different is enough to conclude that $0\text{-}\mathcal{P}_v^{\ell,(i)} \neq 0\text{-}\mathcal{P}_u^{\ell,(i)}$.

For (ii), we prove that PATH-WL is more expressive than 1-WL. This is accomplished by showing an instance of non-isomorphic graphs which 1-WL fails to distinguish but PATH-WL is able to distinguish (see Figure C.1 for one such example). \square

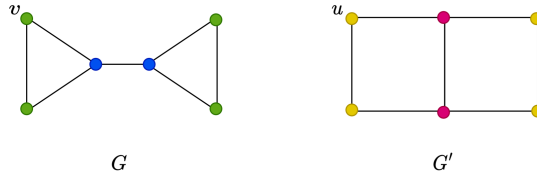


Figure C.1: The coloring after one iteration of PATH-WL is enough to distinguish the two non-isomorphic graphs that 1-WL cannot distinguish. Note that the partition is the same but the colors of the nodes are different.

C.4.2 Proof of Theorem 5.4

In order to prove Theorem 5.4 we need to demonstrate first the following Lemma.

Lemma C.1. *Let C_ℓ be a cycle of length $\ell = \lfloor \frac{k}{2} \rfloor k$. Then C_ℓ has hereditary tree-width $k - 1$.*

Proof. Recall that a graph G has hereditary tree-width $k - 1$ if the maximal treewidth among the graphs in the set of homomorphic images of G is $k - 1$ (See Definition C.3). Thus, in order to prove the lemma, we have to show that (i) there exists a graph H with treewidth $k - 1$, which is a homomorphic image of C_ℓ and also that (ii) there does not exist a different graph H' which is a homomorphic image of G with treewidth larger than $k - 1$.

- (i) For the first part of the proof, we claim that H is a k -clique. Note that the tree-width of a k -clique (cf. Definition C.2) corresponds to the vertices' degree, that is $k - 1$. It remains to prove that a k -clique is a homomorphic image of the cycle C_ℓ , with $\ell = \lfloor \frac{k}{2} \rfloor k$, that is, $\ell = \frac{k-1}{2}k$ for odd k and $\ell = \frac{k^2}{2}$ for even k .

First, consider k to be odd (and therefore $k - 1$ is even). Then, from *Euler's Theorem*, which states that a connected graph has an Eulerian cycle if and only if every vertex has even degree ([193]), we can conclude that a k -clique, with k odd, contains an Eulerian cycle. Based on that, we claim that we can map the Eulerian cycle of a k -clique to the k -clique itself, via a surjective homomorphism ϕ (cf. Definition C.1). This corresponds to prove that the k -clique is the homomorphic image of a cycle C_ℓ (in this case, C_ℓ coincides with the Eulerian cycle). In particular, the number of edges of the Eulerian cycle¹ is equal to the number of edges of the clique, that is $\ell = k \cdot \frac{k-1}{2}$. To guarantee that ϕ is a homomorphism, the vertices in the cycle must be connected in a way that for every edge (u, v) in the cycle, there exists a corresponding edge $(\phi(u), \phi(v))$ in the clique. This is ensured by the definition of Eulerian cycle, which traverses the edges of the clique exactly once. See Figure C.2 for an example of such a homomorphism for $k = 5$.

The resulting function ϕ is surjective because is mapping a cycle of $\frac{k-1}{2}k$ vertices to a clique of k vertices. Hence, we proved the claim for k odd.

Now consider the case of an even k . We aim to show that a k -clique is the homomorphic image of some cycle C_ℓ , with $\ell = \frac{k^2}{2}$. Imagine then, to remove a vertex from the k -clique (as shown in Figure C.3a for the case $k = 6$). Deleting a node results in a $(k - 1)$ -clique that we know from the previous point, is the homomorphic image of its Eulerian cycle. The following procedure illustrates how to build the cycle which will be homomorphically mapped to the k -clique.

- Begin with the cycle on $\frac{k-2}{2}(k - 1)$ edges, the Eulerian cycle of the $(k - 1)$ -clique (See Figure C.3b). We want to add to the cycle the edges that correspond to the missing edges in the k -clique (cf. the dotted edges in Figure C.3a).

First, consider the minimal walk in the cycle, which starts at one node and ends in the same node. This corresponds to "open" the cycle, doubling one node, and leaving the edges fixed. See for example Figure C.4, where we doubled the node a . To preserve the homomorphism, the new node will be mapped to $\phi(a)$.

- As a second step, we need to add to the cycle the edge corresponding to the $k - 1$ edges linking the deleted node x to all the other nodes. We want to do it using the minimal number of edges. Each node in the cycle has degree 2 but x has $k - 1$ neighbors, which is odd. Therefore we will connect x to couples of neighbors (until $k - 2$) and we will have a neighbor repeated twice. In order to utilize the minimum number of edges, the neighbor that will be repeated twice is the node that has been

¹Note that in a cycle, the number of edges corresponds to the number of vertices.

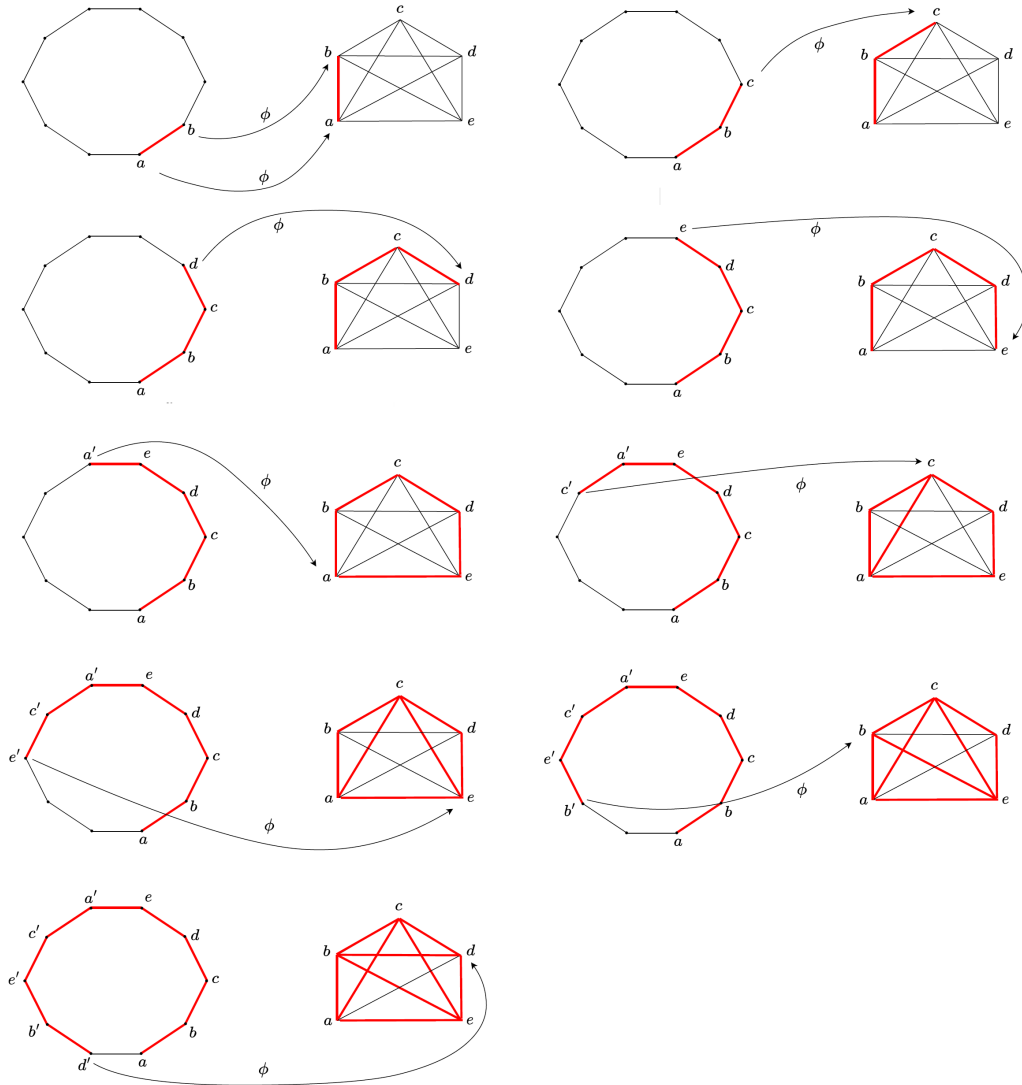


Figure C.2: The construction of the homomorphism ϕ from the cycle C_{10} to the 5-clique. The arrows connect the vertices via the function ϕ , which is surjective. The fact that the nodes are connected such that ϕ is a homomorphism, is guaranteed by the presence of the Eulerian cycle in the clique. Indeed, to construct the cycle it is enough to follow the Eulerian cycle on the clique, starting from any edge.

doubled in the previous step (that is a)². The number of copies of x will be $\frac{k}{2}$ and therefore the number of added edges will be $(2 \cdot \frac{k}{2})$, because with x we add 2 neighbors at a time. It remains to link the neighbors together, with $(\frac{k}{2} - 1)$ edges. The total

²Of course, we can obtain a valid surjective homomorphism by creating a double copy of a node different from a but it will result in the addition of more edges.

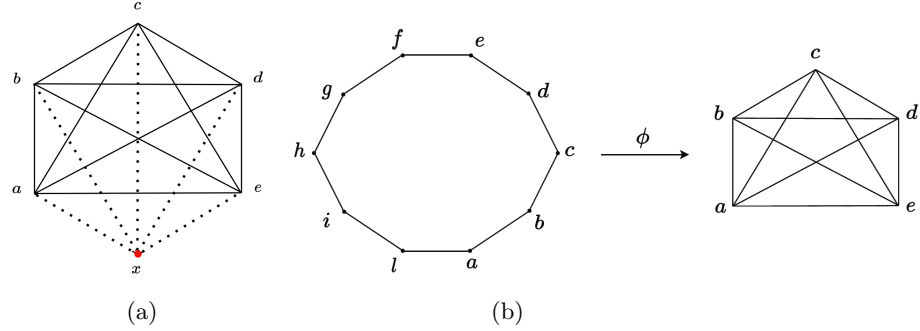


Figure C.3: (a) Deleting one vertex from the 6-clique, the red vertex x , results in a 5-clique. (b) A 5-clique is the homomorphic image of a 10-cycle (See C.1 (i), for k odd).

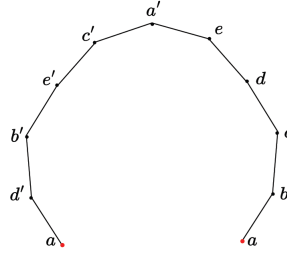


Figure C.4: First step of the procedure: double the red node a and open the cycle. The new node is mapped to $\phi(a)$.

amount of edges in the cycle will be:

$$(k-1) \frac{k-2}{2} + k + \left(\frac{k}{2} - 1\right) = \frac{k^2}{2}$$

(ii) What remains to show is that there does not exist a homomorphic image of C_ℓ with a treewidth larger than $k-1$. For any homomorphic image of C_ℓ on k nodes, we know that the treewidth cannot be larger than $k-1$. Hence, the only possibility to obtain a homomorphic image with a larger treewidth would be to increase the number of vertices (to at most $\lfloor \frac{k}{2} \rfloor k$). However, any homomorphic image with more than k vertices will have a treewidth of at most $k-1$. We prove this based on the concept of forbidden minors: If a graph G does not have K_{n+1} as minor, then G has a treewidth of at most $n-1$. Since C_ℓ by definition only has $\lfloor \frac{k}{2} \rfloor k$ edges, no homomorphic image of C_ℓ can have a K_{k+1} as a minor, and thus the hereditary treewidth of the C_ℓ is $k-1$. □

Now we can finally state and prove Theorem 5.4.

Statement. Let $d \geq 1$ and $k \geq 3$. Then, d -PATH-WL and k -WL are incomparable. Equivalently, the following holds:

- (1) for every $k \geq 1$ there exists a path length ℓ such that d -PATH-WL $^\ell \not\sqsubseteq k$ -WL;
- (2) for every $\ell \geq 1$, there exists a k such that k -WL $\not\sqsubseteq d$ -PATH-WL $^\ell$.

Proof. To prove the first part of the statement, we make use of a recent result from [139, Theorem 1.3]. This result states that for every graph F such that $\text{hdtw}(F) > k$, the k -WL algorithm fails to detect subgraph counts of the pattern F . In our case, let F be a cycle on $\lfloor \frac{k+2}{2} \rfloor (k+2)$ nodes. From Lemma C.1, we know that the hereditary tree-width of a cycle on $\lfloor \frac{k+2}{2} \rfloor (k+2)$ nodes is $k+1 > k$. Then, from Corollary 5.7, we assert that for every $d \geq 1$, d -PATH-WL $^\ell$ can count cycles of any length. In particular, the path length needed to count the pattern F is $\ell = \lfloor \frac{k+2}{2} \rfloor (k+2) - 1$. That is, to distinguish after the first iteration of d -PATH-WL $^\ell$, two graphs indistinguishable by k -WL, we need path length $\ell \sim k^2$.

For the second part of the statement, we show that for every path length ℓ , we can always construct two graphs that are d -PATH-WL-indistinguishable but distinguishable by k -WL for $k \geq 3$. Consider the following graph construction (inspired by the proof for [190, Theorem 6.3]). We set $\mu := 2\ell + 1$ and construct two graphs in the following way: $C_{2\mu}$, which is a cycle of length 2μ , and $C_{\mu,\mu}$, which consists of two disconnected cycles of length μ each. Next, we set $d = \ell$, which aligns with the maximum expressive power of d -PATH-WL $^\ell$. If the two graphs are indistinguishable by d -PATH-WL with $d = \ell$, they are indistinguishable for every $0 \leq d \leq \ell$ (cf. Prop. 5.2). With d -PATH-WL $^\ell$, each node aggregates all the paths of length up to ℓ with shortest path distances $d \leq \ell$. Since each node in $C_{2\mu}$ as well as $C_{\mu,\mu}$ is the starting point for exactly two paths for every length up to ℓ , it will have the same color assignment and the two graphs cannot be distinguished by d -PATH-WL. On the other hand, k -WL can distinguish the two graphs for every $k \geq 3$ ([140, Theorem 6.1]). \square

C.5 Counting cycles

We start by proving the following preliminary Lemma.

Lemma C.2. *Each connected 2-regular graph of n vertices is isomorphic to a cycle C_n .*

Proof. We prove the statement above by induction on the number of vertices. The smallest connected 2-regular graph is a triangle, hence the induction base is $n = 3$. Suppose that $G' = (\mathcal{V}', \mathcal{E}')$ is a connected 2-regular graph with $n + 1$ vertices. Consider one vertex $v \in \mathcal{V}'$, which, by definition, has degree two and therefore two neighbors denoted by v_1 and v_2 . v_1 and v_2 each have another neighbor u_1 and u_2 respectively, with $u_1 \neq u_2 \neq v$, and we thus exclude that they are connected to each other. Indeed, if this were the case, G' would have a disconnected component with 3 vertices. Let G be the graph obtained by removing vertex v from G' and connecting (v_1, v_2) with a new edge (See Figure C.5). The resulting G is connected, 2-regular with n vertices hence, for the inductive hypothesis, it is isomorphic to a cycle on n vertices C_n . If G is a cycle, then adding back the node v and connecting it to v_1 and v_2 as well as deleting the edge (v_1, v_2) is equivalent to adding a path of length two to the cycle. In this way, we obtain a cycle of length $n + 1$, which is isomorphic to G' . \square

C.5.1 Proof of Theorem 5.5

Statement. Let $\text{sub}(C_\ell, G, v) \neq \text{sub}(C_\ell, H, u)$ for some graphs G, H , nodes u, v and cycle C_ℓ . Then, $u \not\approx_{1\text{-PATH-WL}^{\ell-1}} v$.

Proof. The cycle C_ℓ is identified by a path of length $\ell - 1$ where the last node is a marked node. That is, a cycle corresponds to paths of the form: $((\mathbf{c}_{v_1}, 0), \dots, (\mathbf{c}_{v_\ell}, 1))$. Due to the different cycle count at the nodes v and u , their multisets of paths $1\text{-}\mathcal{P}_v^{\ell-1}$ and $1\text{-}\mathcal{P}_u^{\ell-1}$, will contain a

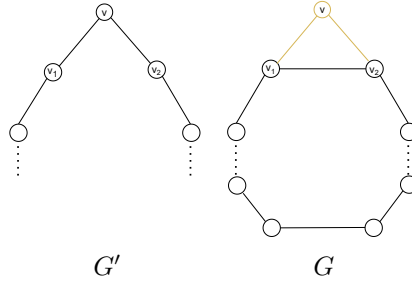


Figure C.5: Sketch for the proof of Lemma C.2.

different number of such paths. That is, $1-\mathcal{P}_v^{\ell-1} \neq 1-\mathcal{P}_u^{\ell-1}$ and given the injectivity of the **Hash** function, also the outcome of 1-PATH-WL will be different for the two nodes. \square

C.5.2 Proof of Corollary 5.8

Statement. For every $\text{expGNN} \in \{\text{SubgraphGNN}, \text{Local 2-GNN}, \text{Folklore } k\text{-GNN}\}$, and for every $k \geq 1$, there exists a path length ℓ such that $1\text{-PATH-WL}^\ell \not\sqsubseteq \text{expGNN}$.

Proof. The proof is a direct consequence of the fact that 1-PATH-WL^ℓ can count arbitrary cycles given a sufficient path length ℓ . In particular, with path length ℓ , 1-PATH-WL can count cycles on $\ell + 1$ nodes, while SubgraphGNN, Local 2-GNN, and Folklore 2-GNN are limited to count cycles on maximum 7 nodes ([136]). Moreover, the Folklore k -GNNs are characterized in expressive power by $k\text{-FWL} \equiv (k+1)\text{-WL}$, for every $k \geq 1$ (cf. Prelim.C.2). Hence, given that 1-PATH-WL^ℓ is not bounded by $k\text{-WL}$ (cf. Theorem 5.4) for length $\ell \sim k^2$, this concludes the proof. \square

C.6 One iteration is almost all you need

C.6.1 Proof of Theorem 5.9

Statement. There exists ℓ such that 0-PATH-WL^ℓ can distinguish the following pairs of infinite graph families:

1. Hamiltonian graphs of different orders at node and graph level,
2. Hamiltonian graphs and non-homogenously traceable graphs at graph level, and
3. almost all connected d -regular graphs and disconnected d -regular graphs with n connected components at graph level.

In order to prove 5.9 we prove (1)–(3) separately.

(1) Hamiltonian graphs of different order.

Corollary C.3. Let \mathcal{H}_n and \mathcal{H}_m be two Hamiltonian graphs with n and m vertices, respectively, and $n > m$. For any $v \in \mathcal{H}_n$ and any $u \in \mathcal{H}_m$, it holds

$$v \not\sim_{0\text{-PATH-WL}^\ell} u \quad \forall \ell \geq m.$$

Proof. By definition, a Hamiltonian graph \mathcal{H} is a graph that contains a cycle C including all the vertices of the graph. Therefore, the length of the Hamiltonian path in \mathcal{H}_m is $m - 1$, which is the longest path in the graph, containing all possible nodes. The same argument holds for \mathcal{H}_n , with $n > m$. In particular, we are able to distinguish two graphs \mathcal{H}_n and \mathcal{H}_m by comparing their path multisets, i.e., after the first iteration, $0\text{-PATH-WL}_v^\ell \neq 0\text{-PATH-WL}_u^\ell \quad \forall \ell \geq m$. \square

(2) Hamiltonian graphs and non-homogeneously traceable graphs. The following theorem states that with sufficiently large ℓ , 0-PATH-WL^ℓ can always distinguish between two classes of graphs: non-homogeneously traceable graphs and Hamiltonian graphs.

Theorem C.4. *Let G and H be two graphs of the same order n . G is a non-homogeneously traceable graph and H a Hamiltonian graph. Then,*

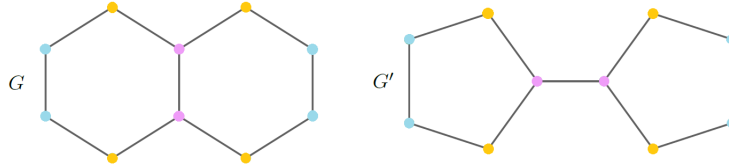
$$G \approx_{0\text{-PATH-WL}^{n-1}} H.$$

Proof. By definition, a non-homogeneously traceable graph is a traceable graph such that at least one node in the graph is not an ending point of a Hamiltonian path. We denote with h_v^{n-1} a generic Hamiltonian path from v . Formally, there exists a node $v \in G$ such that $h_v^{n-1} \notin 0\text{-P}_v^{n-1}(G)$. A Hamiltonian graph is a graph with a Hamiltonian cycle. Any node in the Hamiltonian cycle is an ending point of a Hamiltonian path. Hence, for any node $u \in H$, $h_u^{n-1} \in 0\text{-P}_u^{n-1}(H)$. At graph level, the conclusion follows. \square

Referring to the graph G and G' in figure 5.2, we state that:

Proposition C.5. *For every $\ell \geq 5$, $G \approx_{0\text{-PATH-WL}^{\ell,(1)}} G'$, while $G \sim_{1\text{-WL}} G'$.*

Proof. Let $G = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ and $G' = (\mathcal{V}', \mathcal{E}', \mathbf{X}')$ be the graphs in Figure 5.2. The partition induced on nodes by 1-WL is represented in the following figure, where each color corresponds to a $\sim_{1\text{-WL}}$ equivalence class. As a consequence of Theorem 5.3, the partition induced on the



nodes by 0-PATH-WL is made of three or more classes. Due to the symmetry of the two graphs, we exclude that the nodes can be arranged in more than three equivalence classes, arguing that from symmetric nodes depart the same paths. Hence, to prove the statement is enough to show that the color of one node v in G , via 0-PATH-WL , is different from all the other nodes' colors in G' . Given the injectivity of the **Hash** function, for two nodes to get different colors is sufficient to have different multisets of paths 0-P (this would ensure different colors from the first iteration). We pick the node v in G as in Figure 5.2. Note that $\delta(v) = 3$, and the degree tells us the multiplicity of paths of length 1 in 0-P_v . Therefore, all the nodes u in G' with $\delta(u) = 2$ are such that $0\text{-P}_u \neq 0\text{-P}_v$. Then, we represent in Figure 5.2 the multiset of paths up to length 5, for the node v and the node v' . Due to the different multiplicity of paths of length 5 from the two nodes, we conclude that 0-PATH-WL is able to distinguish the two graphs.

Given that, for every $\ell \geq k$, $0\text{-P}^k \subseteq 0\text{-P}^\ell$, it follows that

$$0\text{-P}_v^k \neq 0\text{-P}_{v'}^k \implies 0\text{-P}_v^\ell \neq 0\text{-P}_{v'}^\ell \quad \forall \ell \geq k.$$

\square

(3) Connected regular graphs vs. disconnected regular graphs. Another property of graphs that is closely related to the concept of paths is *connectivity*. In the following, we prove that paths can distinguish pairs of regular graphs that are 1-WL-equivalent but not isomorphic because one is connected and the other is disconnected. This implies that 1-WL is unable to detect connectivity.

Theorem C.6 (Connectivity). *Let G be a disconnected graph with n connected components which are all d -regular graphs, with $d \geq 2$. Let s be the size of the smallest component. Let G' be a connected d -regular graph, such that $|\mathcal{V}_{G'}| = |\mathcal{V}_G|$. 0-PATH-WL can then distinguish almost every¹ couple of such graphs G, G' , whereas 1-WL cannot. In particular,*

$$G \sim_{WL} G' \quad \text{but} \quad G \not\sim_{0\text{-PATH-WL}^s} G'.$$

Proof. Theorem 1 of [194] states that for $d \geq 3$ almost every d -regular graph is Hamiltonian and from Lemma C.2 we know that every connected 2-regular graph is Hamiltonian. Hence let's consider all the connected components of G and the graph G' as Hamiltonian graphs. Then, from Corollary C.3 we know that two Hamiltonian graphs of different order can be distinguished by 0-PATH-WL ^{ℓ} . The required length ℓ in this case is the size of the smallest connected component of G , i.e. $\ell = s$. Indeed, for every node v in G' , $0\text{-}\mathcal{P}_v^s$ contains a path of length s while $0\text{-}\mathcal{P}_u^s = 0\text{-}\mathcal{P}_u^{s-1}$ for every u in the smallest component of G . Then we can extend the reasoning to the graph level and conclude that $G \not\sim_{0\text{-PATH-WL}^s} G'$. \square

Examples for $d = 2$ and $d = 3$ can be seen in Figure 5.4.

Note that it is always possible to construct a connected d -regular graph by merging two d -regular disconnected components via a *degree-invariant* transformation; See figure C.6 for an example of the transformation process which preserves the degree of the nodes.

Definition C.5 (Degree-invariant transformation). Let $G = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ be a graph. Let $T : \mathcal{E} \rightarrow \mathcal{V} \times \mathcal{V}$ be a transformation on the edges of G , such as the deletion and/or the creation of edges. The transformation is said to be *degree-invariant* if changing the edges does not change the degree of the nodes.

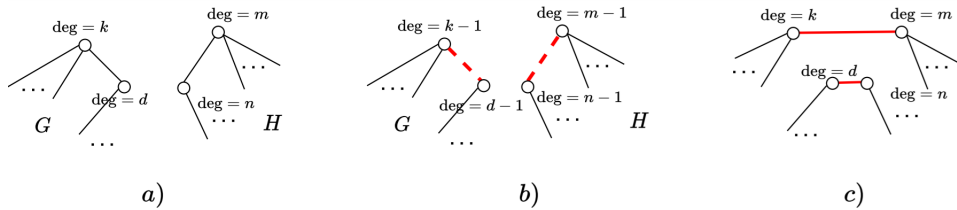


Figure C.6: Example of a degree-invariant transformation. (a) The two graphs G and H can be any two graphs, with at least one edge. (b)–(c) represent the two steps of the transformation: first remove one edge and then link together the two structures in such a way that the degree of the nodes is preserved.

¹Almost all d -regular graphs of order n having a property P means $\lim_{n \rightarrow \infty} Pr(P) = 1$; refer to [194] for details.

C.6.2 Proof of Theorem 5.11

Statement. 1-WL-equivalence at iteration ℓ and 0-PATH-WL $^{\ell,(1)}$ -equivalence are incomparable for every $\ell \geq 3$.

Proof. We prove the following equivalent formulation of the statement. There exist nodes u, v such that for some $k \in \mathbb{N}$

$$u \approx_{0\text{-PATH-WL}^{k,(1)}} v \quad \text{and} \quad u \not\approx_{1\text{-WL}} v \quad \text{at iteration } k.$$

and there exist nodes u', v' such that for some $t \in \mathbb{N}$

$$u' \approx_{0\text{-PATH-WL}^{t,(1)}} v' \quad \text{and} \quad u' \approx_{1\text{-WL}} v' \quad \text{at iteration } t.$$

In order to prove the theorem it suffices to show two examples: (i) one example, where 0-PATH-WL is able to discriminate between two nodes but 1-WL fails, and (ii) a second example where the converse holds. For the first example, we can choose any graph class described before, or the famous instance shown in Figure 5.2. For the other direction, we refer to figure C.7. The highlighted nodes in the figure have different unfolding trees (hence, $u \approx_{UT^3} v$) but they are indistinguishable by the multiset of paths, at least for length $l = 3$.

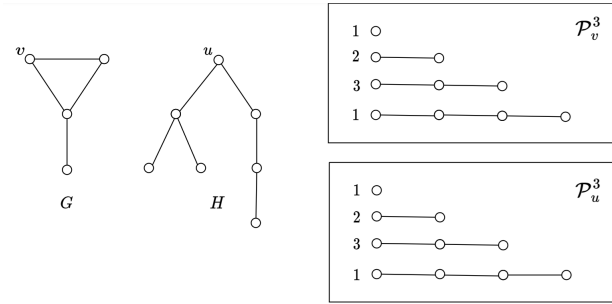


Figure C.7: Graphs G and H and the relative multisets of paths of length up to 3, for the node v in G and u in H . These two graphs serve as a counterexample for paths being more discriminative than unfolding trees (and thus 1-WL).

□

Counterexample for [130, Theorem 3.3] In the following Figure C.8 we computed the path-based unfolding trees **PT** (cf. [130, Definition 3.1]) for the two nodes v in G and u in H .

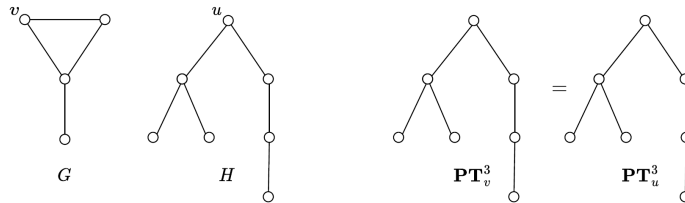


Figure C.8: Counterexample showing that **PT**-equivalence is not more expressive than 1-WL. Indeed $v \approx_{WL} u$ but the path-based unfolding trees for u and v are identical.

C.7 Importance of iterations

In the following, we want to provide some intuition on the importance of the iterations for improving expressivity. For instance, consider the two graphs G and H in Figure C.9. The two highlighted nodes v and u are 0-PATH-WL³-equivalent, as they have the same multiset of paths of length up to 3 (see Figure C.7). Hence, the output of the first iteration of 0-PATH-WL will result in the same color for u and v , as shown in Figure C.9, $it=1$. If we perform another iteration, we can distinguish the two nodes, see Figure C.9, $it=2$.

The iterative procedure increases the expressive power of paths, as colors are assigned to the nodes. The coloring helps to discriminate more between the nodes. Notably, with iterations, we observed that we may reduce the required path length to distinguish two non-isomorphic graphs. For example, in Figure C.10, we can see that the two graphs, which are indistinguishable with one iteration of 0-PATH-WL³, can be distinguished with two iterations of 0-PATH-WL², modifying the complexity from $O(\Delta^3)$ to $O(\Delta^2)$.

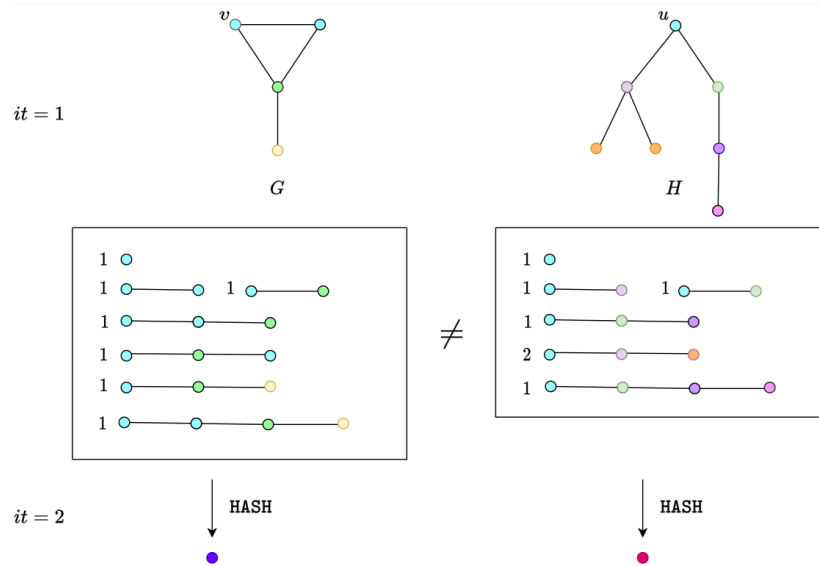


Figure C.9: The two nodes v and u are indistinguishable by 0-PATH-WL^{3,(1)}, indeed they have the same color after iteration 1. But computing the paths with colored nodes allows us to distinguish v and u . That is, $0\text{-PATH-WL}_v^{3,(2)} \neq 0\text{-PATH-WL}_u^{3,(2)}$

C.8 Additional Information on Experiments

Experimental Setup For EXP, SR and ZINC we use a two-layer LSTM for f and summation for graph-level (READOUT) and path-level pooling (AGG). For ZINC we use a shared LSTM for f to reduce the number of parameters. The dataset ZINC also contains categorical node features which we project to the embedding dimension. For ZINC, we incorporate edge features by embedding them and then concatenating them to the vector embeddings in each path. For example in a path (v_1, v_2, v_3) we would attach to the embedding of v_2 the embedded feature of edge (v_1, v_2) and to v_3 the feature of (v_2, v_3) . As the first node v_1 has no associated edge we simply concatenate

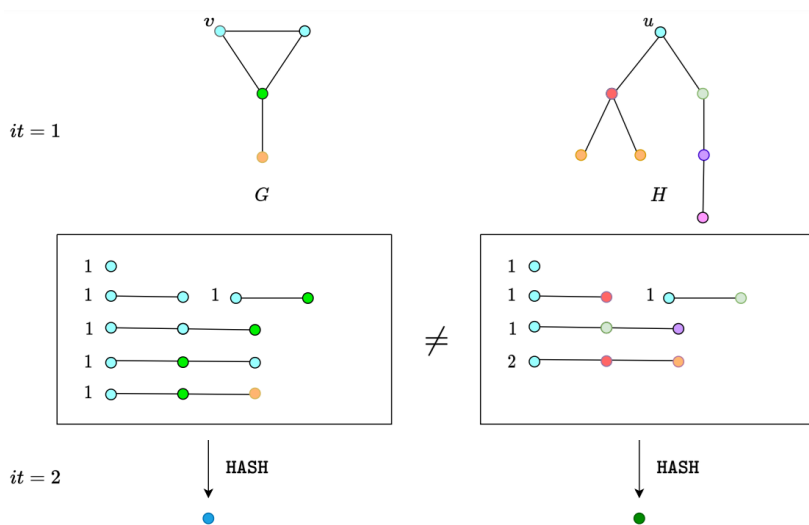


Figure C.10: The two graphs are colored from the first iteration of 0-PATH-WL with paths of length 2. The nodes u and v are not distinguishable, but the multiset of paths contains different paths. The output of the second iteration will result in different colors. That is, $0\text{-PATH-WL}_v^{2,(2)} \neq 0\text{-PATH-WL}_u^{2,(2)}$

an all-zeros vector to it. Finally, we reverse the paths on ZINC as this gives us a small boost in performance (this has also been noticed by [130]).

Due to numerical instabilities encountered with CSL, we vary some parameters for this dataset to combat these instabilities. For CSL we use a one-layer LSTM for f and the mean function for graph-level (READOUT) and path-level pooling (AGG). For CSL and ZINC, we add a two-layer multi-layer-perceptron (MLP) with ReLU activations and batch norm after each LSTM. We use PyTorch ([195]) and PyTorch Geometric ([196]). We conduct our experiments on servers equipped with an RTX-3080 GPU and Intel Core i7-10700KF/i9-11900KF CPU.

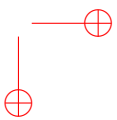
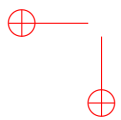
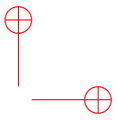
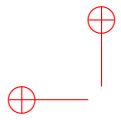
Our code can be found at <https://github.com/ocantias/ExpressivePathGNNs> and <https://github.com/tamaramagdr/synthetic-pain>.

Training Time Comparison. In practice, it can be computationally infeasible to compute all paths. However, as current research on the expressivity of GNNs commonly focuses on molecules that have sparse structures this means that path-based GNNs such as our PAIN can be run efficiently on such datasets. We benchmark the run-time of PAIN on ZINC against GIN [26] and two subgraph GNNs: DS and DSS [151]. For all four models, we select hyperparameters that yield the best validation set performance. For this, we train all four models on ZINC and report the mean and the standard deviation of the training time of each model. For DS and DSS we use a policy that extracts the 3-hop neighborhood (egonets) of every node. PAIN is identical to the model described in Section 5.4. All other models use an embedding dimension of 256 with a dropout rate of 0.5. GIN uses four message-passing layers, and both DS and DSS use five layers. All models are trained with early stopping. Table C.1 shows that PAIN with path length three is only slightly slower than DS and DSS on ZINC (12k graphs) on consumer grade hardware. Finally, we would like to point out that it is not necessary to store every path up to

length ℓ . For the sake of expressivity, it suffices to only store paths that are not part of larger paths. In preliminary experiments, this improved the runtime of PAIN by a speed-up factor of two.

Table C.1: Average training time for different GNNs on ZINC. All models were trained with early stopping.

Model	Average Training Time (\downarrow)
GIN [26]	0.11 ± 0.02 h
DS [151]	0.62 ± 0.06 h
DSS [151]	0.69 ± 0.06 h
PAIN (ours)	0.97 ± 0.09 h





Acknowledgements

Questo percorso durato più di tre anni mi ha fatto crescere tanto e conoscere tante persone che sento il bisogno di ringraziare.

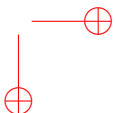
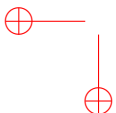
Prima di tutto ringrazio i miei advisor e mentori, Monica Bianchini, Moreno Falaschi e Franco Scarselli. A Monica dedico questo lavoro e in generale questo percorso che senza di lei non sarebbe mai iniziato. All'inizio è stata quasi materna, adesso mi dice "io non voglio soldi, non voglio beghe, fa' tutto te!"

Moreno, esempio di umiltà e colui che ha fatto (e probabilmente farà) le mie presentazioni più lusinghiere. Grazie per la tua considerazione e fiducia.

Franco è probabilmente quello che mi ha insegnato di più, avendo scelto di lavorare con le sue GNN. Formalmente non è il mio advisor ma il suo amore per la ricerca non si ferma di fronte alle formalità. Grazie Franco. Noi tutti abbiamo la fortuna di lavorare con lui e soprattutto di godere della sua risata contagiosa. In un ambiente sereno e familiare, come quello del Lab202.

Amici di lab, vi voglio bene. Non ci sono solo colleghi, sempre pronti a darti una mano, ma ci sono persone che creano legami veri, a ritmo di aperitivi, biliardistico e conferenze. Non vedo l'ora di partire per la nostra vacanza a caso.

Nicco, il mister Wolf che ti fa stare tranquilla e che è sempre pronto a far festa. La conferenza mancata alle Canarie ancora mi brucia. Pietro, se non ci fosse andrebbe inventato. Senza di voi il Monasterino sarebbe stato (ancora più) invivibile. Frank, un ragazzo d'oro, bravo programmatore e pizzaiolo alla sagra della pizza e della pecora. Però questo fatto che la pizza con la pecora non la fate non mi torna. Sari, matematica vera, ormai una certezza dai tempi delle Parabole Grasse. Sopravvissuta con stile all'Alabama verace e a seminari improvvisati di notte. Giova, lei c'è sempre quando si tratta di offrire casa (anche al mare) per far festa, cantare, suonare, e partire insieme! Numero 1. Paolo, da guru dell'informatica ad amico vero, sempre presente in ogni traferta: a Vienna, parte degli (s)tromsi e di avventure australiane. Promessa del biliardo. Grazie per avermi sopportato anche in notti amare come quelle di Melbourne. Marco, Elia, Filippo, siete stati una scoperta a Barcellona. Marco ed Elia mi fate sempre schiantare dal ridere, voi e le vostre Ambre. Pippo, da quando mi hai insegnato "a respirare" riesco a stare meglio, e mi mancano un po' le nostre discussioni davanti a un Margarita. Asma, sei un esempio per me e per tutti noi! Vorrei avere un terzo della tua determinazione e la metà della tua voglia, sei un mito. Barbs, come ho dormito bene con te, mai nella vita. Anche se te non puoi dire lo stesso! Ti stimo tanto, spero che un giorno tu riesca a darti la metà del valore che hai. Fiamma,



arrivata da poco ma già parte integrante del lab! Ti adoro. Cate, Ben, Neda, a voi del primo anno dico, andate in giro più che potete, la ricerca si fa conoscendo persone, scambiando idee, ampliando la vostra rete... e di persona è tutta un'altra cosa!

Ale e Vero, siete stati i miei due compagni di merende... Abbiamo iniziato insieme e siamo stati davvero fortunati a trovarci. Ale, con te Cambridge è un ricordo speciale. Vero, con te anche i viaggi Rigomagno-Siena sono ricordi speciali! E come dimenticare Las Marinas. Vi voglio bene, a ognuno di voi.

Per la mia siamese, non basterebbe tutta la tesi. Grazie a te questo percorso sarà per sempre impresso nel mio cuore. Sono orgogliosa di te e di quanto tu sia stata coraggiosa. Ovunque andrai, sappi che io sarò qui, all'altro capo di un lungo messaggio vocale.

Pensandoci bene, chissà che avrei fatto di questo dottorato se non fossimo mai partiti per Kassel...

Alice and Silvia, my German collaborators and friends from Kassel, I love you. Silvia, you were the smartest and most sensible woman I would have had close to me last YAMC (I was so stressed!) Alice, you're the most brilliant and the funniest girl I have ever met! And a wander mum, now. Hope to see you soon (in Vilnius/Rome)!

Thanks to Alice, I met the people from Vienna. She just threw me towards them saying "She is my smart friend Caterina who is looking for a place to host her for a visiting period!" and that's it. That's the magic of this journey, made of possibilities, research, people, study, people and people again. Like Sagar, to whom I gave a bottle of wine I was hiding... just before he fell down the street... But this is another story. Now I feel like he's a close friend of mine. When I saw him in Vienna, it was clear fresh air. Thanks destiny, thanks God for every single person I met.

I wanna thank Tamara, for our incredible effort and for the patience and perseverance we had, even in bad (French) times:) We did it!

Thanks to Pascal who kindly hosted me the last week in Vienna. Your mattress was incredibly comfy, you made a very good pizza and our party was the best. I would also like to thank my mentor abroad, Thomas Gärtner. You taught me how a paper should be written. Without tiredness in curing every single detail. And also that a glass of good wine can help during hard work nights.

Thanks to Ella, Alex and Jaime: you were my family in Vienna.

Non ci sono parole per ringraziare chi invece mi conosce da sempre, Linda la mia psicologa preferita e Ila, la mia costante. Che a Vienna di fronte a una lavagna con sequenze di triangoli, cuori e cerchi mi chiese, "ma quindi è questo che fai, i disegni?!?" - "Esatto", gli risposi con orgoglio e una risata.

A chi ha condiviso la sua adolescenza con me e ancora non se n'è andato, Rebi e Bene, non ho da dire solo grazie. Rebi, casa vostra è anche casa mia da sempre. Sei l'amica a cui puoi dire tutto, anche quello che non ammetti neanche a te stesso. Bene, con la slitta ci siamo fatte un bel bagno ma anche il pieno di bei ricordi.

Girlz, siete le mie certezze più belle. Grazie per avermi stupito con le vostre cene a sorpresa pre-partenza e per essere semplicemente mie amiche. Vi amo.

Grazie al mio gruppo di danza, per avermi fatto ballare con loro anche tra un viaggio e l'altro, con coreografie imparate sui fogli di carta, accanto alle dimostrazioni. Sul palco insieme siamo ormai una cosa sola.

Tante sono le persone che hanno caratterizzato questo viaggio di 3 anni e mezzo... ma una costante, dietro le quinte, sei sempre stato te, Lori. Ti ringrazio perché mi rendi felice ogni giorno, perché mi sostieni anche quando per te è sacrificio e privazione. Ti ringrazio per aver

amato Vienna anche più di me e per rendere tutto maledettamente facile. Ti amo.

Grazie ai miei geme, vi voglio bene come a due fratelli, ci sarò sempre per voi.

Berni, il mio fratellone, a te, alla mamma e al babbo va il ringraziamento più importante. Ogni mio passo e piccolo successo è per voi e grazie al vostro supporto, di cui avrò sempre bisogno. Vi amo.

Alla mia nonna Givi, che è l'ancora della mia esistenza, Grazie. Con te ho imparato la cosa più importante e il segreto di ogni successo: fare una cosa alla volta, e fare quello che si può. Dopodiché essere soddisfatti di quello che abbiamo ottenuto. E sorprendersi, aggiungo io, a fare più di quello che avremmo immaginato.

Guardando indietro, a quando questo percorso è iniziato (e a quanto fossi convinta di smettere) non mi sembra vero di avere l'entusiasmo che ho oggi. E mi sento fortunata ad avervi avuto vicini, ognuno di voi. Grazie.

Ad Maiora!

Cate

