

Extending Graph Neural Networks with Global Features

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Data Science

eingereicht von

Dragos-Andrei Brasoveanu, BSc

Matrikelnummer 12045444

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ. Prof. Dr. Thomas Gärtner

Mitwirkung: Maximilian Thiessen

Wien, 2. September 2024

Dragos-Andrei Brasoveanu

Thomas Gärtner



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Extending Graph Neural Networks with Global Features

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Data Science

by

Dragos-Andrei Brasoveanu, BSc

Registration Number 12045444

to the Faculty of Informatics

at the TU Wien

Advisor: Univ. Prof. Dr. Thomas Gärtner

Assistance: Maximilian Thiessen

Vienna, September 2, 2024

Dragos-Andrei Brasoveanu

Thomas Gärtner



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Erklärung zur Verfassung der Arbeit

Dragos-Andrei Brasoveanu, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 2. September 2024

Dragos-Andrei Brasoveanu



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Acknowledgements

I would like to start by thanking my advisor, Maximilian Thiessen, for his countless hours of work, support and great feedback, and for his exceptional dedication in helping me complete this thesis, even in the final, last-minute stages.

Special thanks to Fabian Jogl and Pascal Welke, for helping me with code repository and experiment setup management, ideas for the thesis and invaluable last-minute feedback.

I want to thank Prof. Thomas Gärtner for helping and giving me the opportunity to work on my thesis within his department.

I would also like to thank my friends for their unwavering support and love. A special thanks to Mihai and Ioana for always helping me stay focused and motivated to finish this thesis.

Last and the most important, I would have not been able to finish this thesis without the unconditional support and love from my family during the last three years. Their constant encouragement and belief in me provided the strength I needed to get where I am today.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Kurzfassung

Wir untersuchen die Verwendung globaler Graphmerkmale, um die Ausdruckskraft und die Vorhersageleistung von graph neural networks (GNNs) zu verbessern. Während Message Passing Neural Networks (MPNNs) ein gängiger Ansatz für das Lernen von Graphrepräsentationen sind, sind sie in ihrer Ausdruckskraft beschränkt. Diese Einschränkung beeinträchtigt die Universalität von MPNNs und folglich die Arten von Grapheneigenschaften, die sie darstellen können.

Um diese Herausforderungen anzugehen, schlagen wir vor, expressive globale Graphmerkmale in eine Basis-GNN-Architektur zu integrieren. Im Gegensatz zu herkömmlichen Ansätzen, die die Vorhersageleistung entweder durch Änderungen im Message Passing oder durch Modifikation der Graphen (z. B. Hinzufügen zusätzlicher Knotenmerkmale) verbessern, konzentriert sich unsere Methode auf globale Graphmerkmale, die die gesamte Graphstruktur beschreiben. Diese globalen Merkmale, die in Bereichen wie der Chemoinformatik gut etabliert, jedoch in der GNN Literatur oft übersehen sind, können verwendet werden, um wichtige Aspekte des Graphen zu erfassen, die von MPNNs nicht ausgedrückt werden können, wie etwa den Graphendurchmesser oder den längsten Zyklus.

Wir ergänzen die von GNNs gelernten Embeddings mit ausgewählten globalen Graphmerkmalen und deren Einfluss auf die Ausdruckskraft und Modelleistung. Wir zeigen, dass 19 der 23 untersuchten globalen Merkmale, wie der Wiener- oder der Hosoya-Index, nachweislich die Ausdruckskraft von MPNNs erhöhen. Unsere empirischen Studien belegen, dass die Integration globaler Graphmerkmale die Vorhersagekraft von GNNs bei molekularen Benchmark-Datensätzen verbessern kann, was auf eine vielversprechende Richtung zur Steigerung der Effektivität von GNN-basierten Modellen hinweist.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Abstract

We investigate the use of global graph features to enhance the expressive power and predictive performance of graph neural networks (GNNs). While message passing neural networks (MPNNs) are a common approach for learning graph representations, they are inherently limited in their expressive power, which restricts their ability to distinguish between certain graph structures. This limitation impacts the universality of MPNNs, constraining the range of functions they can approximate and, consequently, the types of graph properties they can detect.

To address these challenges, we propose a method for incorporating expressive global graph features into any baseline GNN architecture. Unlike traditional approaches that enhance predictive performance by either changing the message passing or modifying the graphs (e.g. attaching additional node features), our method focuses on global graph properties that describe the entire graph structure. These global features, well-established in fields like chemoinformatics but often overlooked by the GNN community, can be used to capture important aspects of the graph not expressible by MPNNs, such as the graph diameter or the longest cycle.

We propose to extend the generated embeddings learned by GNNs with selected global graph features, analyzing their impact on expressiveness and model performance. We show that 19 out of the 23 global features we have investigated, like the Wiener or the Hosoya index, provably increase the expressivity of MPNNs. Our empirical studies demonstrate that incorporating global graph features can improve the performance of GNNs on molecular benchmark datasets, suggesting a promising direction for increasing the effectiveness of GNN-based models in graph-level tasks.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Contents

Kurzfassung	ix
Abstract	xi
Contents	xiii
1 Introduction	1
2 Preliminaries	5
2.1 Graphs	5
2.2 Machine Learning on Graphs	9
2.3 Training Neural Networks	10
2.4 Message Passing Neural Networks	16
2.5 Representational power of Message Passing Graph Neural Networks	17
3 Related Work	19
3.1 Graph Invariants	19
3.2 Molecular Descriptors	20
3.3 Expressive Power of Graph Neural Networks	22
4 GNNs with Global Features	23
4.1 Motivation	23
4.2 Global Features	24
4.3 Expressivity of GFs	32
4.4 GNNs with Global Features	40
5 Experiments	43
5.1 Experimental Setup	43
5.2 Results and Discussion	46
6 Conclusion	53
List of Figures	55
List of Tables	57
	xiii

List of Algorithms

57

Bibliography

59

Introduction

In this thesis, we investigate the limitations of graph neural networks (GNNs) in terms of their expressive power for distinguishing different graphs. It has been proven that message passing graph neural networks (MPNNs) have limited expressive power with respect to distinguishing graphs [Xu et al., 2019; Morris et al., 2019]. A common approach to making GNNs more expressive is to modify the message passing algorithm underlying MPNNs. While this method can yield more expressive GNNs, it has the drawback of requiring custom message passing implementations that may not be compatible with existing algorithms. In order to overcome this downside and to increase the generalizability of the GNNs used, another approach is to include a preprocessing stage on the graphs, by applying graph transformations or enriching the node features. The advantage of this method is that it can be easily combined with existing GNNs implementations, as it only requires changes to the input graph data.

Moreover, it has been proven that standard message passing models cannot compute certain graph properties that are global in nature, e.g., the graph diameter, the longest cycle in a graph. Following this direction, we want to study whether adding this type of information, i.e., graph invariants describing the whole graph structure, can improve the expressiveness of GNNs and increase the empirical performance on graph-level tasks. We want to integrate these global properties with the generated embeddings of the GNN, so that (1) they do not require any changes to the GNN implementation and (2) they can be benchmarked across a wide variety of GNN classes, e.g., GIN, GCN or CWN. We show that combining global features with MPNNs can increase their expressivity in relation to the Weisfeiler-Leman (WL) algorithm. Moreover, we demonstrate that the addition of the global features to the GNNs increases the empirical performance on molecular datasets.

Given this potential to improve GNN expressiveness through global properties, we explore their application in real-world scenarios, such as drug discovery. Graphs naturally represent complex entities like molecules, making it feasible to use graph learning

approaches to predict their behavior (see Figure 1.1). For example, traditional drug discovery processes involve labor-intensive and costly experimentation to determine how different molecules interact with biological targets. Instead, by using GNNs, a high number of molecules can be screened in a shorter time and have their key molecular behaviours predicted, such as binding affinity, toxicity or bioavailability. A critical aspect of this process involves datasets that provide the necessary molecular information for training these models. One such dataset is ZINC [Sterling and Irwin, 2015], containing small-molecules together with their logP values, which is a measure of how soluble a molecule is. Every molecule that is a potential drug candidate needs to go through a process of optimization of this variable, as it impacts its bioavailability. Therefore, by training a GNN that is able to accurately predict logP, several elaborates of such a molecule can be rapidly screened and evaluated *in silico*. This approach saves significant time and resources that would otherwise be spent on synthesizing and experimentally testing each variation in the lab.

While GNNs have demonstrated significant potential in predicting molecular properties, recent research has also highlighted certain limitations regarding their expressive power. For example, Garg et al. [2020] have shown that GNNs based on message passing are inherently unable to compute specific global features of a graph, such as graph diameter or the longest cycle, which are important for capturing the full structural complexity of molecules. Conversely, Huuskonen et al. [1998] introduced an approach that utilizes such global features, also known as *topological indices*, to effectively predict molecular solubility. We propose a novel strategy combining both approaches by integrating global graph features with GNN-learned embeddings. This combined method aims to produce models that (1) are more expressive compared to their base GNN counterpart (with no global features attached) and (2) demonstrate enhanced empirical performance on graph-level tasks, particularly within molecular applications.

Our contributions are as follows:

- We propose attaching the global features together with the generated embeddings of a GNN.
- We select a subset of features that are computed based on different graph properties: distances, node degrees, eigenvalues and number of matchings. By constructing these global features independently, we ensure they can be easily integrated with any existing GNN, regardless of its underlying message-passing algorithm.
- In order to leverage this method, we apply it to three different of GNNs: CWN [Bodnar et al., 2021], GIN [Xu et al., 2019] and GCN [Kipf and Welling, 2017].
- We study the expressiveness of the GNNs combined with the global features and prove that an MPNN combined together with global features can be more expressive than the MPNN alone.

-
- Furthermore, we demonstrate that our method achieves better performance with these GNNs on graph molecular datasets.

Structure of this thesis. We begin by defining the fundamental concepts related to graphs, machine learning on graphs, and training neural networks, including message passing graph neural networks (MPNNs), in Chapter 2. This section also discusses the representational power of MPNNs, providing a thorough background for the subsequent discussions. In Chapter 3, we present an overview of related work, starting with a review of graph invariants and molecular descriptors, and ending in a discussion on the expressive power of different graph neural networks in recent literature. Chapter 4 introduces our approach to incorporating global features with GNNs, beginning with the motivation and a detailed explanation of the global features utilized. We then analyze the expressivity of these global features and demonstrate how they can be integrated into GNN models. In Chapter 5, we describe our experimental setup and present the results together with a discussion, demonstrating the effectiveness of our proposed method by benchmarking it against various molecular datasets. Finally, Chapter 6 concludes this work by summarizing our findings and suggesting directions for future research.

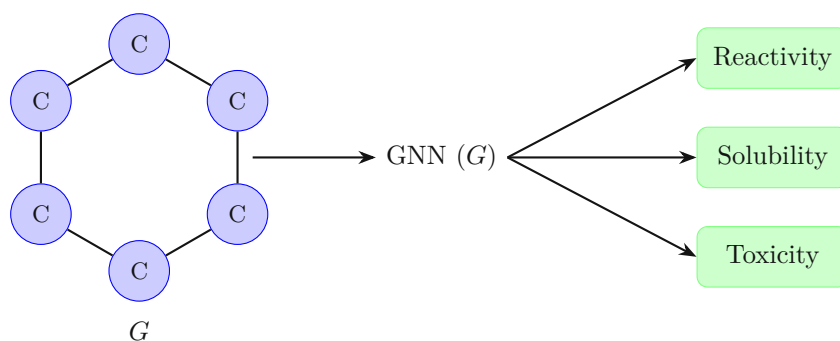


Figure 1.1: A benzene molecule represented as graph G (ignoring hydrogen atoms), where the nodes represent the atoms of carbon and the edges the bonds between the respective atoms. The GNN can predict certain molecular behaviours in relation to different media: reactivity, solubility and toxicity.

Part of this thesis has been published as:

- Andrei Dragos Brasoveanu, Fabian Jögl, Pascal Welke, and Maximilian Thiessen. Extending graph neural networks with global features. In *The Second Learning on Graphs Conference*, 2023.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Preliminaries

In this section, we introduce the basic concepts needed for understanding expressiveness in the context of graph theory and its applications within graph neural networks (GNNs). We begin by introducing graphs (Section 2.1) and then give a short introduction on supervised machine learning (Section 2.2). We then go on to briefly outline the methodologies involved in training deep neural networks (Section 2.3) and describe one important family of GNNs, message passing neural networks (Section 2.4), along with its representational capabilities in the context of expressivity (Section 2.5).

2.1 Graphs

We start by defining the concept of a graph and introducing several important concepts related with it. The definitions are based on the book of Diestel [2005] and adapted based on the style of notations in the GNN literature.

We define a graph $G = (V, E)$ to be an ordered pair of a set of vertices (or nodes) V and a set of edges E , which are themselves pairs of vertices. We denote by \mathcal{G} the set of all graphs. Formally, we define $G = (V, E)$, where:

- V is the set of vertices of graph G , $V = \{v_1, v_2, \dots, v_n\}$,



Figure 2.1: Left: a complete graph with 4 nodes, denoted as K_4 , right: a graph in the shape of a hexagon, denoted as C_6 , resembling a benzene molecule.

- E is the set of edges of graph G , $E = \{e_1, e_2, \dots, e_m\}$.

For the purpose of our work, we only consider graphs G where $V(G)$ is non-empty and finite. In the case of a graph representation of a molecule, an edge represents a mutual direction between two atoms, without implying a direction of the interaction. Consequently, our work will focus the class of undirected graphs, thus the set of edges $E(G)$ contains unordered pairs of vertices $E(G) \subseteq \{\{v, w\} \mid v, w \in V(G), v \neq w\}$. Each edge $e_i = \{v, w\} \in E$ connects two different vertices v, w ; we say that v and w are *neighbours* (or *adjacent*) and that v and w are *incident* with edge e_i . For this purpose, we define the *degree* of a vertex, denoted $\deg(v)$, as the number of edges incident to v ; it can be calculated as $\deg(v) = |\{e \in E : v \in e\}|$. We say that $v \in V(G)$ is said to be a *leaf* vertex if $\deg(v) = 1$. Furthermore, we denote $\mathcal{N}_G(v) = \{u \in V(G) \mid (u, v) \in E(G)\}$ the *neighbourhood* of node v , i.e., the set of nodes adjacent to node v . We use the $\{\{\cdot\}\}$ notation to denote a *multiset*.

For a graph G with n vertices and an ordered vertex set $V = \{v_1, \dots, v_n\}$, the *adjacency matrix* is an $n \times n$ matrix, where each element can be defined by:

$$A_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \in E \\ 0 & \text{otherwise} \end{cases}. \quad (2.1)$$

We say that the *degree matrix* D of graph G is an $n \times n$ *diagonal* matrix, i.e., a matrix for which all of its non-diagonal elements are equal to 0, where each element is defined as:

$$D_{ij} = \begin{cases} \deg(v_i) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}. \quad (2.2)$$

A *complete* graph with n vertices, denoted K_n , is a graph in which every pair of distinct vertices is connected by a unique edge. An example of a complete graph with 3 vertices K_3 (called a triangle) can be seen in Figure 2.1 on the left.

A *path* in a graph is a non-empty graph $P = (V, E)$ where the vertex set $V = \{x_0, x_1, \dots, x_k\}$ is ordered such that each pair of consecutive vertices is connected by an edge in the edge set $E = \{x_0x_1, x_1x_2, \dots, x_{k-1}x_k\}$, and all vertices x_i are distinct. The *length* of the path is the number of edges, denoted by $|E| = k$. Additionally, we define a *cycle* as a graph C formed by a path $P = x_0, x_1, \dots, x_{k-1}$ together with an edge connecting the last vertex x_{k-1} back to the first vertex x_0 , where $k \geq 3$ is the minimum length. The *length* of the cycle is equal to its number of vertices.

The *distance* $d_G(x, y)$ in G of two vertices x, y is the length of a shortest x - y path in G ; if no such path exists, we set $d(x, y) := \infty$. The *graph distance* matrix, sometimes also called the all-pairs shortest path matrix, is the square matrix $(d(i, j))$ consisting of all graph distances from vertex v_i to vertex v_j [Graham and Pollak, 1971].

We define the *Laplacian* matrix of graph G following Chung [1997]. Given an unweighted graph G , the Laplacian matrix $L_{n \times n}$ is defined element-wise as:

$$L_{ij} := \begin{cases} \deg(v_i) & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise} \end{cases} \quad (2.3)$$

One can use an equivalent definition of the Laplacian matrix using the adjacency matrix A and the degree matrix D :

$$L = D - A. \quad (2.4)$$

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs. If $V_2 \subseteq V_1$ and $E_2 \subseteq E_1$, then G_2 is a *subgraph* of G_1 (and G_1 are a *supergraph* of G_2), written as $G_2 \subseteq G_1$. We say that G_1 and G_2 are *isomorphic*, denoted by $G_1 \simeq G_2$, if there exists a bijection $f : V_1 \rightarrow V_2$ with $\{v_1, v_2\} \in E_1$ if and only if $\{f(v_1), f(v_2)\} \in E_2$, for all $v_1, v_2 \in V_1$ (see Figure 2.2), e.g. this means that any two vertices v_1, v_2 are adjacent in G_1 if and only if $f(v_1), f(v_2)$ are adjacent in G_2 . Such a function f is called an *isomorphism*.

Given two graphs $G = (V, E)$ and $H = (V', E')$, *subgraph isomorphism* amounts to finding a subgraph $G_0 = (V_0, E_0)$ of G such that:

- $V_0 \subseteq V$, and $E_0 \subseteq E \cap (V_0 \times V_0)$,
- $G_0 \simeq H$, i.e., there exists a bijection $f : V_0 \rightarrow V'$ such that:

$$\{v_1, v_2\} \in E_0 \iff \{f(v_1), f(v_2)\} \in E'.$$

Babai [2016] defines the *graph isomorphism problem* to be to determine whether two given finite graphs are isomorphic or not. There is no known to be any algorithm so far that solves this problem in polynomial time, nor is it known to be an *NP-complete* problem. Thus, there is no efficient solution at the moment that solves this problem. However, Babai [2016] shows that it can be solved in quasipolynomial ($\exp((\log n)^{O(1)})$) time.

Weisfeiler-Leman test We will now give an overview of a standard algorithm used to test a pair of graphs for isomorphism: the *Weisfeiler-Leman graph isomorphism test* [Weisfeiler and Leman, 1968]. For this purpose, we define the concept of *labeled graphs* (e.g., see Chartrand and Zhang [2008]). We define a *labeled graph* as $G = (V, E, \ell)$, such that $\ell : V \rightarrow \mathcal{L}$ is a labeling function that maps each vertex to a label from the set of labels \mathcal{L} (which could be the set of positive integers \mathbb{N}). We denote $\mathcal{G}_{\mathcal{L}}$ the set of all labeled graphs.

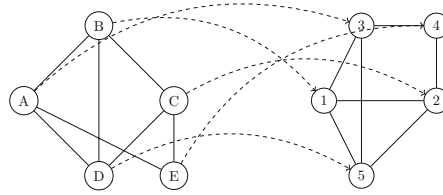


Figure 2.2: A pair of isomorphic graphs (dotted lines show the mapping between the nodes of the two graphs).

We formally define the Weisfeiler-Leman (WL algorithm) algorithm. It is also known as a *color refinement* algorithm. Given a labeled graph G with a node coloring ℓ , an iteration of the WL algorithm will produce a new node coloring ℓ' . We use the $c_i(v)$ to denote the color of vertex v at iteration i .

1. Assign all vertices the same color $c_0(v), v \in V$.
2. For a vertex v in iteration $i + 1, i > 0$ its color is defined by

$$c_{i+1}(v) = \text{HASH}(c_i(v), \{\{c_i(u) : u \in \mathcal{N}_G(v)\}\}),$$

where HASH is an injective hash function that maps to \mathcal{L} (typically \mathbb{N}).

3. Continue iterating until the partition of vertices induced by the colors remains stable, i.e., when the sets $\{v \in V : c_{i+1}(v) = \alpha\}$ for each color α do not change between iterations i and $i + 1$.

The algorithm can be defined then as $\text{WL}(G) = \{\{c_{\text{final}}(v) : v \in V(G)\}\}$, where $c_{\text{final}}(v)$ is the labeling of the vertices after the algorithm stabilizes. WL is guaranteed to terminate within at most $|V(G)|$ iterations [Jegelka, 2022].

To determine whether two graphs G_1 and G_2 are isomorphic or non-isomorphic, the WL algorithm is applied to both graphs. After stabilization, the final label multisets of G_1 and G_2 are compared:

- If the final label multisets are different, then G_1 and G_2 are *not* isomorphic.
- If the final label multisets are identical, then G_1 and G_2 *might* be isomorphic. However, this result is inconclusive because the WL algorithm is not a complete test for graph isomorphism—it can fail to distinguish between some non-isomorphic graphs. Further testing or different methods would be needed to confirm isomorphism in this case.

2.2 Machine Learning on Graphs

We now introduce the basic concepts of machine learning applied on graphs. For this purpose, we follow the definitions and notations used in Jegelka [2022]. We extend our definition of a graph as an ordered pair $G = (V, E)$ to $G = (V, E, X, W)$, where $X \in \mathbb{R}^{d \times |V|}$ is the matrix containing the d -dimensional attribute vector for each node (*node feature matrix*) and $W \in \mathbb{R}^{d_w \times |E|}$ is the matrix containing the d_w -dimensional attribute vector for each edge (*edge feature matrix*). Note that X and W may be empty. We use the notation of $X(v)$ to represent the feature vector of node v and $W(v, u)$ to represent the feature vector of edge $\{u, v\}$.

Let \mathcal{Y} be a label set and \mathcal{G} the set of all graphs. Let \mathcal{F} be a set of models such that for any $F \in \mathcal{F}$, we have $F : \mathcal{G} \rightarrow \mathcal{Y}$, i.e., a function that, when applied to a graph $G \in \mathcal{G}$, computes a label $y \in \mathcal{Y}$. In this context, it follows that any graph neural network (GNN), which takes a graph as input and produces a label as output, can be viewed as an element of the class \mathcal{F} . For the purpose of our work, when learning a GNN, we observe N independent and identically distributed (i.i.d.) samples $\mathcal{D} = \{G_i, y_i\}_i^N \in (\mathcal{G} \times \mathcal{Y})^N$ drawn from an underlying distribution \mathcal{P} on $\mathcal{G} \times \mathcal{Y}$. Given a loss function $\ell : \mathcal{G} \times \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$ that measures the prediction error, that is, the discrepancy between y and $F(G)$, our aim is to estimate a model $F \in \mathcal{F}$ that minimizes the loss function:

$$\min_{F \in \mathcal{F}} \mathbb{E}_{(G, y) \sim \mathcal{P}} [\ell(G, y, F(G))].$$

Note that, even with sufficient approximation power, we can only estimate a function $\hat{F} \in \mathcal{F}$ from \mathcal{D} , and since the distribution \mathcal{D} is unknown, the common procedure for *training* is to minimize the empirical risk:

$$\hat{F} \in \arg \min_{F \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^N \ell(G_i, y_i, F(G_i)).$$

In practice, a learned model \hat{F} is usually evaluated on data that is drawn from a different distribution $\mathcal{Q} \neq \mathcal{P}$ to the one that the samples \mathcal{D} were drawn from, e.g., graph of different sizes, node or edge feature value ranges. We usually work with a training and test set: the learning process is performed on the training set and we evaluate the estimated models on the test set (unseen data for the estimated model). The goal is to minimize the mismatch between the estimations done by the model, $F(G)$, and the labels y .

In the context of this thesis, we focus on graph-level tasks, where the goal is to predict properties or labels associated with entire graphs $G \in \mathcal{G}$, rather than individual nodes or edges. Below are some representative examples of graph-level tasks:

1. **Molecular Property Prediction:** A common application in chemoinformatics involves predicting the properties of molecules, such as toxicity, solubility, or binding affinity. Here, each molecule is represented as a graph $G = (V, E, X, W)$, where the

nodes V represent atoms and the edges E represent chemical bonds. The task is to predict a continuous or categorical label $y \in \mathcal{Y}$ for the entire molecule based on its structural features captured in the node feature matrix X and edge feature matrix W .

- Social Network Analysis:** In social networks, graphs $G = (V, E, X)$ represent individuals as nodes V and relationships or interactions as edges E , with node features X potentially capturing individual attributes. A graph-level task might involve predicting whether a given social network exhibits certain properties, such as containing influential communities or having a high degree of connectivity. The entire network is treated as a single unit, and the goal is to assign a label $y \in \mathcal{Y}$ to it accordingly.
- Computer Vision:** In some image recognition tasks, images can be represented as graphs $G = (V, E, X)$, where superpixels or regions of interest are treated as nodes V and their spatial relationships are represented by edges E , with node features X describing pixel or region properties. The task might involve classifying the entire image based on these graph representations, for example, determining whether an image contains a specific object or scene, resulting in a label $y \in \mathcal{Y}$ for the graph.

To summarize, we have a list of graphs and their corresponding labels. Our goal is to estimate a function $F \in \mathcal{F}$ which minimizes the mismatch of label y and prediction $F(G)$. A concrete example of a graph level (regression) task is predicting the constrained solubility of a molecule [Irwin and Shoichet, 2005]. The set of labels for this given task is $\mathcal{Y} = \mathbb{R}^+$ and one possible goal is to minimize the difference $|y - F(G)|$.

Permutation Invariance. In graph-level tasks, such as the ones mentioned above, it is crucial that the learned function F is independent of how the graph is represented. Since the nodes of a graph can be arbitrarily indexed, F should yield the same output regardless of any permutation of the node ordering. For example, for the molecular property prediction task mentioned above, the order of atoms should not affect the predicted property (e.g., toxicity or solubility) of the whole molecule. This requirement, known as *permutation invariance*, ensures that the predictions depend solely on the graph’s structure and features, rather than on the particular ordering of its nodes, thereby making the model robust and applicable across varying graph representations.

Definition 1. Let $f : \mathcal{G} \rightarrow \mathbb{R}^n$ be a function. The function f is said to be permutation invariant if, for any pair of graphs $G_1, G_2 \in \mathcal{G}$ such that $G_1 \simeq G_2$, it holds that:

$$f(G_1) = f(G_2).$$

2.3 Training Neural Networks

In this section we will cover the basic concepts involved in training (*deep*) neural networks ((D)NNs). For this, we use Goodfellow et al. [2016] as our main guidance. This section

serves the purpose of covering fundamental concepts in machine learning which will be later used to explain the message passing neural networks (MPNNs). This section assumes the context of euclidean space data, i.e., instead of graphs, the training data is represented as $(x_i, y_i)_{i=1}^N$, where $x_i \in \mathbb{R}^n, y_i \in \mathcal{Y}$. We start by defining the *multilayer perceptron*, the quintessential deep learning model. We then explain how the deep neural networks are trained and why is it difficult to apply this framework to a non-euclidean data space directly, such as graph data.

Multilayer Perceptron (MLP). The multilayer perceptron is one of the most important and basic deep learning models. Its goal is to approximate some function f^* such that $y = f^*(x)$ is a mapping of an input x to a label y . Such a model defines then a mapping $\mathbf{y} = f(\mathbf{x}, \boldsymbol{\theta})$ and then learns the value of the parameters $\boldsymbol{\theta}$ which result in the best approximation of the function. We use the term *network* in this context because the models are typically represented as a composition of many functions, connected as an acycling graph, in a chain, e.g., $f(\mathbf{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x})))$. The term used to describe each function $f^{(i)}$ is a *layer*, i.e., $f^{(1)}$ represents the first layer of the network. The length of the chain determines the *depth* of the model and the last layer is called the *output layer*. Each layer allows for parallel transformations of its input data, using the *learned* parameters and followed by a *non-linear activation* function. Such a transformation can be defined as [Goodfellow et al., 2016, p.172]:

$$\mathbf{h} = g(\mathbf{W}^\top \mathbf{x} + \mathbf{c}), \quad (2.5)$$

where \mathbf{x} is the input data fed into the first layer of the MLP. Any subsequent layer will use as input the computation result of the previous layer $h^{(i-1)}$. A more specific definition of a MLP with j layers then is as follows:

$$\begin{aligned} h^{(1)} &= g^{(1)}(\mathbf{W}^{(1)\top} \mathbf{x} + b^{(1)}), \\ h^{(i)} &= g^{(i)}(\mathbf{W}^{(i)\top} h^{(i-1)} + b^{(i)}), \quad \forall i \in \{2, \dots, j\} \end{aligned} \quad (2.6)$$

For every layer i in this definition, $h^{(i)}$ represents the output of i -th layer of the network and $\mathbf{W}^{(i)} \in \mathbb{R}^{n_{i-1} \times n_i}$ is the *weight matrix* associated with the i -th layer, where n_{i-1} and n_i represent the *width* of the $(i-1)$ -th and i -th layer, i.e., the number of units (or *neurons*) in each layer. $g^{(i)}$ represents the non-linear activation function of layer i , applied to the linear transformation of the input $\mathbf{W}^{(i)\top} h^{(i-1)} + b^{(i)}$, where $b^{(i)} \in \mathbb{R}^{n_i}$ is the *bias* vector associated with the i -th layer. In modern neural networks, a common choice for the non-linear activation function g is to use the *rectified linear unit* or ReLU, defined as [Goodfellow et al., 2016, p.168-p.175]:

$$\text{ReLU}(x) = \max\{0, x\}.$$

The output of such a network will be the computation result of the last layer j , $z = h^{(j)} \in \mathbb{R}^{n_j}$. Since the network provides a set of hidden features $f(\mathbf{x}; \boldsymbol{\theta})$, the role of the

output layer should then be to provide some additional transformation to fit the task that the overall network must perform. Given a regression task that aims to predict a continuous numeric value from a one-dimensional feature space, then $z \in \mathbb{R}$, e.g., predict the future price of a stock based on historical data. However, for a *binary classification* problem, where we have to predict the value of a *two-class* variable y , an output consisting of a vector of real numbers is not appropriate. For this purpose, we need to apply a transformation to convert output z into a probability $P(y = 1|\mathbf{x})$ that lies between 0 and 1, such that [Goodfellow et al., 2016, p.182-183]:

$$\hat{y} = \sigma(z),$$

where $\hat{y} = P(y = 1|\mathbf{x})$, $\hat{y} \in [0, 1]$ is the predicted probability and σ is the *logistic sigmoid* function defined as [Goodfellow et al., 2016, p.67]:

$$\sigma(x) = \frac{1}{1 + \exp(-x)}.$$

In the case of a *multiclass* problem, one would have to represent the probability distribution over m different classes (where m is the number of possible classes). Thus, we generalize the problem of the binary classification such that we produce a vector $\hat{\mathbf{y}}$, with $\hat{y}_i = P(y = i|\mathbf{x})$, where $\hat{y}_i \in [0, 1]$, $\sum_{i=1}^m \hat{y}_i = 1$. For this purpose, one can use the *softmax* function, given by [Goodfellow et al., 2016, p.184-185]:

$$\text{softmax}(z)_i = \frac{\exp(z_i)}{\sum_{k=1}^{n_j} \exp(z_k)}. \quad (2.7)$$

Losses. We define the loss function $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}^+$, such that $\mathcal{L}(\hat{y}, y)$ quantifies the cost associated with predicting label \hat{y} when the ground truth is label y . One common type of loss function, primarily used for regression tasks is the *mean squared error*. It measures the averages of the squares of the errors, i.e., the average squared difference between the estimated values and the actual value, defined as [Goodfellow et al., 2016, p.108]:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2, \quad (2.8)$$

where n is the number of samples in the input data.

For a classification problem, one could use the *cross-entropy* loss, defined for multi-class classification tasks. It is applied when the model outputs a probability distribution across multiple classes for each instance in the input data. It is defined as [Bishop, 2006, p.209]:

$$-\sum_{i=1}^N \sum_{c=1}^M y_{ic} \log(\hat{y}_{ic}), \quad (2.9)$$

where N is the number of samples in the input data, M is the number of possible classes, y_{ic} is a binary value (0 or 1) indicating whether class c is the correct label for instance i and \hat{y}_{ic} is the predicted probability of instance i being of class c . This definition assumes that the y_{ic} are provided as a *one-hot* encoded vector per instance, i.e., each instance has exactly one class labeled as 1 and all others 0.

Optimizers. As it was mentioned in the MLP section, the goal of the network is to learn the parameter vector θ which results in the closest approximation of a function which maps input x to label y . Optimizers solve this problem, by minimizing the loss function $\mathcal{L}(\theta)$ using an iterative updating process. A general form of an update rule for an optimizer can be expressed as [Bishop, 2006, p.240]:

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla \mathcal{L}(\theta_t), \quad (2.10)$$

where θ_t is the parameter vector at iteration t , $\eta > 0$ is known as the *learning rate*, a positive scalar determining the size of the update for each iteration, and $\nabla \mathcal{L}(\theta_t)$ is the gradient of the loss function with respect to the parameters at θ_t . The value of θ is set to some starting vector θ_0 . Note that the loss function is defined with respect to a input data set (e.g., training set), and so each iteration requires that the entire set be processed to evaluate $\nabla \mathcal{L}$. We call such an iteration an *epoch*. In each iteration, the parameter vector θ is moved in the direction of the greatest rate of decrease of the error function, so this approach is known as *gradient descent*.

Such methods that use the whole input data at once are called *batch* methods. If the input dataset is sufficiently large, one might consider a *sequential algorithm*, in which the data points are considered one at a time and the parameter vector θ gets updated after each such presentation. An example of such an algorithm is the *stochastic gradient descent* (SGD), defined as Bishop, 2006, p.240:

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla \mathcal{L}_n(\theta_t). \quad (2.11)$$

As the learning rate η determines the size of the step, adapting it during the training process is desirable. A few factors that are to be considered for the choice of η are:

- the size for the input data batch (also known as *batch size*): a sequential gradient estimator such as the SGD may introduce noise due to the random sampling of the training samples, compared to a batch gradient descent where the input set of training samples stays the same across different iterations
- information contained within the gradient of the loss function

In practice, it is necessary to decay η over time. The primary concern is determining the appropriate initial setting for the learning rate η_0 . If set too high, the learning curve

may show large oscillations, and the cost function could increase substantially. Mild oscillations are generally acceptable, particularly when training involves a sequential optimizing algorithm, such as the SGD. Conversely, if the learning rate is set too low, the learning process will proceed slowly. A very low initial learning rate might even lead to the learning stalling at a high cost value. Typically, the optimal initial learning rate, considering both total training duration and final cost value, is higher than a rate that would produce the best performance within the first 100 iterations. [Goodfellow et al., 2016, p.295]

Recently, a number of optimization methods to adapt the learning rates of the model parameters have been introduced, such as Duchi et al. [2011] or Kingma and Ba [2017]. They will not be discussed in detail as their intricacies are not in the scope of our work.

Schedulers. The different parameters discussed above, which have a high influence on the training results, such as the learning rate, batch size or the width of a MLP layer are called *hyperparameters* [Goodfellow et al., 2016, p.98]. The learning rate is perhaps the most important hyperparameter - the effective capacity of the model is the highest when the learning rate is correct for the optimization problem, not when it is at a very low or very high value. Depending on the stage of the training, the learning rate should be set to different values. Nowadays, it is common to use a decay scheme, based on different strategies [Goodfellow et al., 2016, p.425-429]:

- linear decay until reaching a minimum preset value for the learning rate
- exponential decay
- decreasing by a custom factor, e.g., 2-10, each time a metric (such as the error on the validation set) plateaus, e.g., *ReduceLROnPlateau*¹

Dataset split. In order to evaluate the performance and *generalizability* of a model, we need to evaluate the learnt parameter vector θ on a set of data points that were held-out from the training dataset, but coming from the same distribution [Goodfellow et al., 2016, p.121]. For this purpose, we partition the input dataset \mathcal{D} into three subsets:

- training set \mathcal{D}_{train} : used to train the model and adjust the parameter vector θ
- validation set \mathcal{D}_{val} : used to estimate the generalization error *during* training, allowing for the hyperparameter set to be updated accordingly
- test set \mathcal{D}_{test} : used to evaluate the model's performance after the training and the hyperparameters tuning processes are complete

¹PyTorch version https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.ReduceLROnPlateau.html

Typically, one uses 80% of the input training data set for training and 20% for validation [Goodfellow et al., 2016, p.121].

One problem with partitioning the dataset into a fixed train and test set may be that the resulting test set is too small, which results in statistical uncertainty of the generalization error measured on the test data. One solution to this problem is to obtain partitions by splitting the dataset into k non-overlapping subsets, a procedure known as *k-fold cross-validation*. The overall generalization error of the model may be then estimated by averaging the test error across the k partitions. In partition i , the i -th subset of the input data is used as the test dataset and the rest is used as the training data [Goodfellow et al., 2016, p.122].

Bias-variance tradeoff. Once the training process is finished, we need to evaluate its performance. The factors which determine how well does the model perform are its ability to:

- make the training error small (reduce *bias*)
- make the gap between training and test error small (reduce *variance*)

These two factors correspond to two important problems in machine learning: *underfitting* and *overfitting*. Underfitting occurs when a model is not able to obtain a sufficiently low error on the training set, which may happen due to a low *capacity* of the model, i.e., a low variety of functions that the model can fit. For example, the capacity could be increased by adding more neurons or layers to a model. However, models with too high of a capacity may overfit by learning highly specific properties of the training set that are not found within the test set [Goodfellow et al., 2016, p.111-112]. In order to prevent overfitting, besides decreasing the model's capacity, one could use *regularization* techniques, such as *dropout* or *early stopping*.

Regularization. Regularization can be informally defined as any modification that we make to a learning algorithm that has the purpose of reducing its generalization error (test set error) but not its training error [Goodfellow et al., 2016, p.119].

Early stopping. When training models with enough capacity to be able to overfit the task, it is often observed how the training error steadily decreases over time, while the validation error begins to rise again. This behaviour occurs reliably and is a sign of a model that starts to overfit the training data. In order to obtain a model with a better generalizability, we should return the parameter setting at the point in time where the validation error was at its lowest. Every time we observe an improvement in the validation set error, we store a copy of the model parameters. When the training process terminates, we return these parameters, rather than the latest parameters. The training process will be stopped when no set of parameters that improve the validation

set error (compared to the stored parameters) have been found for a pre-specified number of iterations [Goodfellow et al., 2016, p.246-247].

Dropout. Another method of regularization that is both powerful and computationally inexpensive is *dropout*. It involves training the ensemble that consists of all sub-networks that can be formed by removing non-output units, i.e., neurons from the last layer. With most modern neural networks, one can effectively remove a unit from a network by multiplying its output value by zero. When training a model with dropout, we use a minibatch-based learning algorithm, such as SGD. Every time we load a sample into a minibatch, we randomly sample a different *binary mask* to be applied to all of the input and hidden units of the network. The probability of sampling a mask value of one, i.e., the corresponding masked unit should be included in the network, is a hyperparameter set before the training begins. Usually, the probabilities set for the input units and hidden units is set to 0.8 and 0.5 respectively [Goodfellow et al., 2016, p.258-259].

Universal Approximation Theorem. One might assume that learning a nonlinear function requires designing a specific model for the kind of nonlinearity we want to learn. Fortunately, feedforward networks with hidden layers, which include MLPs, provide a universal approximation framework. The *universal approximation theorem* [Hornik et al., 1989] asserts that a feedforward neural network, equipped with a linear output layer and at least one hidden layer is capable of approximating any continuous function on a closed and bounded subset of \mathbb{R}^n , between two finite-dimensional spaces. This approximation can achieve any desired non-zero amount of error, as long as the network has a sufficient number of hidden units, meaning that regardless of what function we are trying to learn, a large enough MLP will be able to *represent* this function. However, it is not guaranteed that the training algorithm will be able to *learn* that function [Goodfellow et al., 2016, p.198].

2.4 Message Passing Neural Networks

MLPs, by processing inputs in a fixed order, do not accommodate the unordered nature of graph nodes. Consequently, the functions learned by MLPs do not inherently exhibit permutation invariance, which is a crucial property for graph-level tasks (as discussed in Section 2.2). One common way to achieve permutation invariance is through the use of message passing neural networks (MPNNs) [Gilmer et al., 2017], which are specifically designed to handle graph-structured data by learning functions that remain invariant under any permutation of the nodes.

In this section we define the *message passing graph neural networks* [Gilmer et al., 2017]. For this purpose, we use the definitions of Jegelka [2022].

MPNNs follow an iterative scheme, throughout which they maintain an embedding $h_v^{(t)} \in \mathbb{R}^{d_t}$ for each node $v \in V$. In each iteration t , the embedding $h_v^{(t)}$ gets updated as a function of its neighbours' embeddings and edge attributes (in case there are any):

$$\begin{aligned}
 h_v^{(0)} &= x_v, & \forall v \in V \\
 m_v^{(t)} &= f_{\text{Agg}}^{(t)}(h_v^{(t-1)}, \{\{h_u^{(t-1)}, w(u, v) \mid u \in \mathcal{N}(v)\}\}), & 1 \leq t < T. \\
 h_v^{(t)} &= f_{\text{Up}}(h_v^{(t)}, m_v^{(t)}), & 1 \leq t < T
 \end{aligned} \tag{2.12}$$

The last node representation $f(v) = h_v^{(T)}$, for all $v \in V$ is the result of the last iteration, which may be concatenated with a linear classifier. Here, $h_v^{(t)}$ encodes the t -hop neighbourhood of node v , i.e., the subgraph consisting of all nodes that may be reached from node v with a path of maximum length t . The number of iterations T can be also used to denote the *depth* of the graph neural network (GNN), and such an iteration may be viewed as a layer.

The *aggregation* function $f_{\text{Agg}}^{(t)} : \mathbb{R}^{d_{t-1}} \rightarrow \mathbb{R}^{d_t}$ is a nonlinear function, shared by all nodes, of the form:

$$f_{\text{Agg}}^{(t)}(h_v^{(t-1)}, \{\{h_u^{(t-1)}, w(u, v) \mid u \in \mathcal{N}(v)\}\}) = \phi_1^{(t)}\left(\sum_{u \in \mathcal{N}_v} \phi_2^{(t)}(h_u^{(t-1)}, h_v^{(t-1)}, w(u, v))\right). \tag{2.13}$$

The sum operation applied on the result of the $\phi_2^{(t)}$ can be replaced by an average, degree-normalized sum or a maximum. The functions $\phi_1^{(t)}$, $\phi_2^{(t)}$ are implemented as MLPs.

The *update* function f_{Up} is of form:

$$f_{\text{Up}}(h_v^{(t)}, m_v^{(t)}) = \sigma(W_1^{(t)} h_v^{(t)} + W_2^{(t)} m_v^{(t)}), \tag{2.14}$$

where σ is a coordinated-wise nonlinear activation function and W_1 , W_2 are learnable weight matrices.

Since the scope of our work is graph-level tasks, all node representations must be aggregated. One could use a permutation invariant *readout* function, as such:

$$F(G) = f_{\text{Read}}(\{\{h_v^{(T)} \mid v \in V\}\}). \tag{2.15}$$

2.5 Representational power of Message Passing Graph Neural Networks

A standard way to characterize the discriminative power of MPNNs is via the graph isomorphism problem [Xu et al., 2019; Morris et al., 2019]. Since the functions generating the embeddings for the message passing algorithm treat multiple nodes as sets or multisets, their order is ignored, i.e., MPNNs are *permutation invariant*.



Figure 2.3: The molecules bicyclopentyl (left) and decalin (right) which cannot be distinguished by WL algorithm and thus by any MPNN. Node colours are the result of the WL node colouring algorithm. Figure adapted from Sato [2020].

Thus, two isomorphic graphs will be mapped to the same output. Said differently, if the mappings generated are different, the graphs are not isomorphic. However, as we have seen with the *WL* algorithm described in Section 2.1, in case two graphs get mapped to the same output, we cannot tell whether they are isomorphic or not (inconclusive result).

In this context, we define *expressivity* to be the ability of a permutation invariant function f to map two non-isomorphic graphs G, H to different embeddings $f(G) \neq f(H)$. We say that a permutation invariant function f_1 is at *least as expressive* as permutation invariant function f_2 if f_1 can distinguish every pair of graphs that f_2 can distinguish. Moreover, if f_1 is at least as expressive as f_2 and can distinguish at least one more pair of graphs than f_2 , we say that f_1 is *more expressive* than f_2 .

Relationship of MPNNs and WL algorithm test. Xu et al. [2019] and Morris et al. [2019] have proven that MPNNs are at most as expressive as the WL algorithm.

Theorem 1 (Xu et al. 2019; Morris et al. 2019). *MPNNs are at most as expressive as the WL algorithm.*

This theorem implies that there exist pairs of non-isomorphic graphs, G_1, G_2 , such that any MPNN F will generate the same result, $F(G_1) = F(G_2)$. Such a pair can be seen in Figure 2.3. This result is a consequence of the upper-bounded *expressive power* of the MPNN and is independent of its underlying architecture or weights, i.e., other variants of MPNNs. This motivates the search for more expressive models.

To show the upper bound of expressiveness for message passing graph neural networks, Xu et al. [2019] introduced the *Graph Isomorphism Network* (GIN). GIN is provably the most expressive MPNN and equally expressive as the WL test, provided it has sufficiently many layers. The node embedding update operation in layer t for GIN is as follows:

$$h_v^{(t)} = \phi^{(t)} \left((1 + \epsilon^{(t)}) \cdot h_v^{(t-1)} + \sum_{u \in \mathcal{N}_G(v)} h_u^{(t-1)} \right). \quad (2.16)$$

In this equation, $\phi^{(t)}$ represents a multi-layer perceptron (MLP), and $\epsilon^{(t)}$ is a scalar that can either be learnable or fixed.

Related Work

In this chapter we give an overview of the related literature. We start by examining works which include the concept of graph invariants (Section 3.1). We then shift our focus to literature that includes the use of topological indices or other molecular descriptors (e.g., molecular fingerprints). Finally, we review works which discuss expressivity and distinguishing power in the context of GNNs (Section 3.3).

3.1 Graph Invariants

In this section, we explore key works in the GNN literature that focus on *graph invariants*. Graph invariants are properties of graphs that remain unchanged under graph isomorphisms. This ability to remain unchanged makes them particularly useful in applications where graph structures need to be compared or classified. Graph invariants for example can be algebraic, combinatorial, or topological in nature and include: the number of vertices, the number of edges, degree sequences, and more complex properties such as the eigenvalues of the adjacency or of the Laplacian matrices.

In network analysis, *spectral invariants* derived from the eigenvalues of graph matrices (such as the adjacency matrix or Laplacian matrix) play a crucial role. These invariants include the *spectrum of a graph*, consisting of the eigenvalues of the adjacency matrix [Von Collatz and Sinogowitz, 1957] or the *algebraic connectivity*, which is the second smallest eigenvalue of the Laplacian matrix of a graph [Fiedler, 1973] and measures the robustness and overall connectivity of a graph. Lim et al. [2023] introduced *SignNet* and *BasisNet*, two neural network architectures invariant to two key symmetries displayed by eigenvectors: sign flips and more general basis symmetries which occur in higher dimensional eigenspaces, where eigenvalues can be added in each layer. They discuss the expressivity of SignNet with and without added eigenvalues, relying on common results concerning Weisfeiler-Leman and spectral graph theory, as shown in Rattan and Seppelt [2023].

More recently, Song et al. [2024] proposed an adaptive multi-view parallel graph contrastive learning framework (*AMPGCL*), which uses global information for graph representation learning. By leveraging a dual-view approach, where feature and topological views are encoded using graph convolutional networks (GCNs) and a transformer to handle multi-hop relationships in graphs, AMPGCL demonstrated superior performance across various graph benchmarks.

Another important class of graph invariants include graph counting features, such as *subgraph patterns* and *homomorphism counts*. Subgraph patterns, such as cliques, cycles, or other small graph motifs, provide insights into the local connectivity and structural properties of larger graphs. Homomorphisms are a more relaxed notion of a subgraph. A *homomorphism* $\Phi : V(F) \rightarrow V(G)$ is a map that preserves edges between the vertex sets, i.e., $\{v, w\} \in E(F) \Rightarrow \{\Phi(v), \Phi(w)\} \in E(G)$. Barceló et al. [2021] proposed a method, termed \mathcal{F} -MPNN, through which the initial features of the vertices are augmented by incorporating local graph parameters, such as homomorphism counts of small graph patterns, while maintaining the same $\mathcal{O}(n)$ memory efficiency. Using the same integration method, i.e., augmenting the original node features, Bouritsas et al. [2023] proposed using subgraph isomorphism counts as additional information, which are then input into the GNN. By contrast, Welke et al. [2023] considers homomorphism counts attached at a graph-level (in contrast to node-level) to improve the GNNs, i.e., attaching them to the generated embeddings by the MPNN. We follow this approach.

Finally, as Garg et al. [2020] have proven, several important graph invariants, e.g., shortest/longest cycle, diameter or certain motifs, cannot be computed by GNNs. Such GNNs include standard message passing models, and more powerful variants that exploit local graph structure (e.g., via relative orientation of messages, or local port ordering) to distinguish neighbours of each node.

3.2 Molecular Descriptors

Building on the concept of graph invariants, which provide a foundation for understanding and comparing graph structures, we now turn to their application in the domain of molecular graphs. In computational chemistry and chemoinformatics, graph-based representations of molecules rely on *molecular descriptors*—quantitative measures that encapsulate the structural properties of a molecule. These descriptors, like graph invariants, must be permutation-invariant to ensure that molecular properties are consistently predicted regardless of the ordering of atoms. Molecular descriptors play a critical role in computational chemistry and chemoinformatics, as they provide *quantitative* representations of molecular structures. This section focuses on results which use two important types of permutation-invariant molecular descriptors: *molecular fingerprints* and *topological indices*.

Molecular fingerprints represent molecules as fixed-length binary vectors, where each bit corresponds to the presence or absence of specific substructures or patterns within the molecular graph. The process of generating a fingerprint involves enumerating various

substructures, such as paths, cycles, or predefined patterns, from a molecular graph $G = (V, E)$. These substructures are then hashed or indexed into a binary vector, ensuring that the resulting fingerprint is invariant under graph isomorphisms.

Rittig et al. [2023] explored the possibilities of GNNs in generating a molecular fingerprint representation through the combination of message passing and pooling operations, to predict molecular structure-property relationships, e.g., the *boiling point* of a given molecule. Alternatively, Wen et al. [2022] introduces a method for molecular property prediction involving a pretrained language model (*BERT* [Devlin et al., 2019]) on molecular fingerprints. The model is then used to generate fingerprints for a given molecule and utilizes a convolutional neural network (CNN) to predict the final molecular properties. In Offensperger et al. [2024], the authors used a large data set (> 250000) of *fully functional fragments* (FFFs) to train a *bespoke* molecular fingerprint for a specific type of compound. The molecular fingerprint was then subsequently used to encode and successfully predict ligand behaviour in cells for a new, unseen set of compounds.

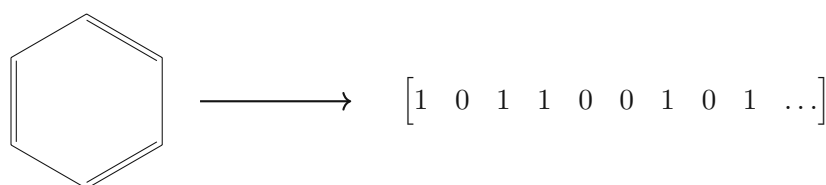


Figure 3.1: Conversion of a benzene ring to a molecular fingerprint.

Molecular fingerprints provide a compact, permutation-invariant representation of molecular graphs by capturing the presence or absence of specific substructures. However, these binary vectors do not capture all the structural properties of a molecule, e.g., distances or node distributions in the graph. To achieve a more detailed characterization of the molecules, another class of molecular descriptors, known as *topological indices*, is used. Topological indices are numerical values calculated directly from the graph's structure, such as the Wiener index [Wiener, 1947] or the Randic index [Randic, 1975], and provide a more comprehensive characterization of the molecular graph by encoding information about the entire graph topology, including aspects such as connectivity, node degree distributions and branching patterns.

Horn et al. [2022] introduced the concept of *Topological Graph Neural Networks* (TOGL), a *layer* which integrates global topological information, such as the *circuit rank* (also known as the first Betti number), using persistent homology. The authors show that TOGL can be integrated into *any* type of GNN whilst simultaneously improving the predictive performance for graph classification tasks. Parveen et al. [2022] explored the use of topological indices in the development of novel drugs through *Quantitative Structure-Property Relationship* (QSPR) modeling. The authors demonstrated that certain physicochemical properties of the compounds, such as refractivity and complexity, could be *linearly* modeled using different topological indices, with these properties showing a strong linear correlation to the indices.

3.3 Expressive Power of Graph Neural Networks

We will analyze our proposed approach, which combines GNNs with global features, through the lens of expressivity. After Xu et al. [2019] and Morris et al. [2019] MPNNs are bounded in terms of expressivity by the WL algorithm, considerable effort has been dedicated to developing more expressive MPNNs and GNNs.

One of the first contributions for more expressive GNNs was done by extending the WL test to k tuples of nodes [Morris et al., 2019]. The k -WL test generalizes the WL algorithm test by considering k -tuples rather than individual vertices, thereby increasing the expressiveness of the test, at the cost of runtime. Morris et al. [2019] introduced k -dimensional GNNs that extend the idea of k -WL to GNNs, such that, for a sufficiently large neural network, k -GNNs are equally as expressive as k -WL is. However, their runtime increases exponentially with k .

Another method to increase the expressivity of MPNNs involves introducing a *pre-processing stage*. In this stage, the initial labeling of the graph, such as vertex features, can be extended with additional information derived from properties of the graph. As we have seen in Section 3.1, Barceló et al. [2021] proposed \mathcal{F} -MPNN, that enriches the initial vertex features with rooted homomorphism counts of small graph patterns, increasing the expressive power of the overall GNN. Similarly, Bouritsas et al. [2023] uses rooted subgraph isomorphism counts as additional information. Most of these pre-processing approaches attach additional information about the graphs on the node level. However, some graph properties are inherently non-local (e.g, eigenvalues of the Laplacian) and so it seems logical to concatenate the same value to all nodes. As this leads to unnecessary redundancy, a more space efficient approach would be to attach the features just once at the global level. An example of this method is seen in the work of Welke et al. [2023], where additional information, specifically homomorphism counts, is incorporated at the graph level with the embeddings learnt by the MPNN. Similarly, Papp et al. [2021] introduced Dropout Graph Neural Networks (DropGNNs), which consider leaving out information from the initial graph for the pre-processing stage. They operate by executing multiple runs of a GNN on the same graph, each time randomly and independently dropping some of the nodes. The results from these runs are then combined to produce the final output. This dropout technique increases the expressive power of the GNNs: even when two distinct d -hop neighborhoods cannot be distinguished by a standard GNN, their dropout variants (with a few nodes removed) are already separable by GNNs in most cases. Thus by learning to identify the dropout patterns where the two d -hop neighborhoods differ, DropGNNs can also distinguish a wide variety of graphs that are beyond WL.

GNNs with Global Features

In this chapter, we introduce the concept of *global features* (GFs). We show that any graph-level global feature can be combined with a base graph neural network (GNN) without requiring modifications to the network itself. In Section 4.2, we define the global features used in our work and propose a method for categorizing them, based on the graph properties that they depend on.

Next, in Section 4.3, we study the expressiveness of global features in relation to the WL algorithm. We demonstrate that there are global features which can distinguish pair of graphs that the WL algorithm cannot.

Finally, in Section 4.4, we present our method of combining GNNs with global features. We show that it is possible to increase the expressive power of a GNN with the addition of a GF.

4.1 Motivation

A common assumption in the design of GNNs is that the predictions made by the model should remain consistent, regardless of the order in which the nodes of the graph are presented. This principle is known as *permutation invariance*. Permutation invariance is crucial because, in many applications, the structure of the graph rather than the specific ordering of its nodes carries the relevant information. As outlined in Section 2.5, GNNs are permutation invariant.

Similarly, the WL algorithm test aggregates and updates node colours between iterations based on the multiset of labels in each node's neighborhood. This mechanism is closely related to the message passing in MPNNs and as a result, the expressivity of MPNNs is inherently upper-bounded by the WL algorithm test [Xu et al., 2019; Morris et al., 2019] (more details in Section 2.5).

Limitations of WL/MPNNs. While the WL algorithm and its related GNN architectures are able to distinguish a high number of classes or types of graphs, they are not sufficiently expressive to capture all relevant aspects of graph structures. Specifically, they struggle with graph properties that are not captured at the node level and are *global* in nature. For many global properties there is no trivial way of *localizing* the graph-level feature to the node-level. For example, it is possible to obtain a local version of the average node degree across the graph: the degree of each node can be computed by summing messages from its neighbors. The average degree of nodes can then be computed by aggregating the node degree information using a readout function like mean pooling. This is not possible, for example, with the eigenvalues of the Laplacian as they are inherently non-local.

One method of integrating this information is to attach these features, either the local or global variant of them, to the nodes or edges, as shown before by Barceló et al. [2021] and Bouritsas et al. [2023]. However, this might dilute the impact of more meaningful node-specific or local features. For example, in a graph where local clustering coefficients are important, attaching a global feature like graph diameter to every node might mask the significance of the local clustering information. Moreover, attaching global feature values to nodes introduces the risk of *oversmoothing*, an issue which occurs when node representations become indistinguishable from each other after multiple layers of aggregation, leading to a loss of unique node-specific information. This can particularly happen with GNNs where iterative aggregation can amplify the uniformity introduced by global features. Attaching this information to the nodes would add redundant information as well and thus it is more appropriate to use such features globally.

The purpose of this work is to address this challenge by utilizing graph permutation-invariant functions, computed at a global graph level, to enhance the embeddings generated by the GNN, without making changes to the GNN architecture. Specifically, we examine existing numerical or ordinal properties that (1) consider various aspects of the graph and (2) have been previously employed in practice. These properties offer a high-level summary of the graph’s structure, topology, or other aggregate characteristics. We refer to these properties as *global features*.

Definition 2. (*Global Feature*) Let f be a permutation invariant function with $f : \mathcal{G} \rightarrow \mathbb{R}$. Then we call f a *global feature*.

4.2 Global Features

In this section we present the global features (GFs) that were selected for this work, along with their classifications and formal definitions. We have picked 23 GFs guided by two primary considerations: first, to cover diverse graph properties such as distances, node degrees, eigenvalues or matchings in the graph. Second, we leverage the *mathchem* Python library [Vasilyev and Stevanovic, 2013]¹, which offers a large collection of topological

¹Code can be found at <https://github.com/hamster3d/mathchem-package>.

indices and other invariants of molecular graphs. A tree diagram illustrating the choice of different global features classified based on the graph properties that they are computed on is provided below (see Figure 4.1).

4.2.1 Degree-based GFs

Degree-based global features are calculated based on the *degrees* of the vertices within a graph. For instance, the *Randic index* is a well-known degree-based index, that was found to correlate with the boiling points of the alkanes and organic molecules. Molecules with higher branching (and thus a lower Randic index) often have lower boiling points due to reduced surface area [Randic, 1975]. It is defined as:

$$R(G) = \sum_{\{u,v\} \in E} \frac{1}{\sqrt{\deg(u) \cdot \deg(v)}}.$$

Besides the Randic index, we select two other features that are computed using the node degrees:

- Zagreb M_1 index [Gutman and Trinajstić, 1972]:

$$M_1(G) = \sum_{v \in V(G)} \deg(v)^2.$$

- Zagreb M_2 index [Gutman and Trinajstić, 1972]:

$$M_2(G) = \sum_{\{u,v\} \in E(G)} \deg(u) \deg(v).$$

4.2.2 Distance-Based GFs

Distance-based global features are derived from the *shortest path lengths* between pairs of vertices in a graph. These indices measure how far apart vertices are from one another, offering valuable information on the overall structure and compactness of the graph. The *Wiener index*, for example, is a prominent distance-based GF that sums the shortest path lengths between all pairs of vertices. It has been shown that it is strongly correlated with the boiling points of alkane molecules [Wiener, 1947]. It is defined as:

$$W(G) = \sum_{\{u,v\} \subseteq V} d(u,v).$$

Besides the Wiener index, we select six other GFs that are computed using the distances in the graph:

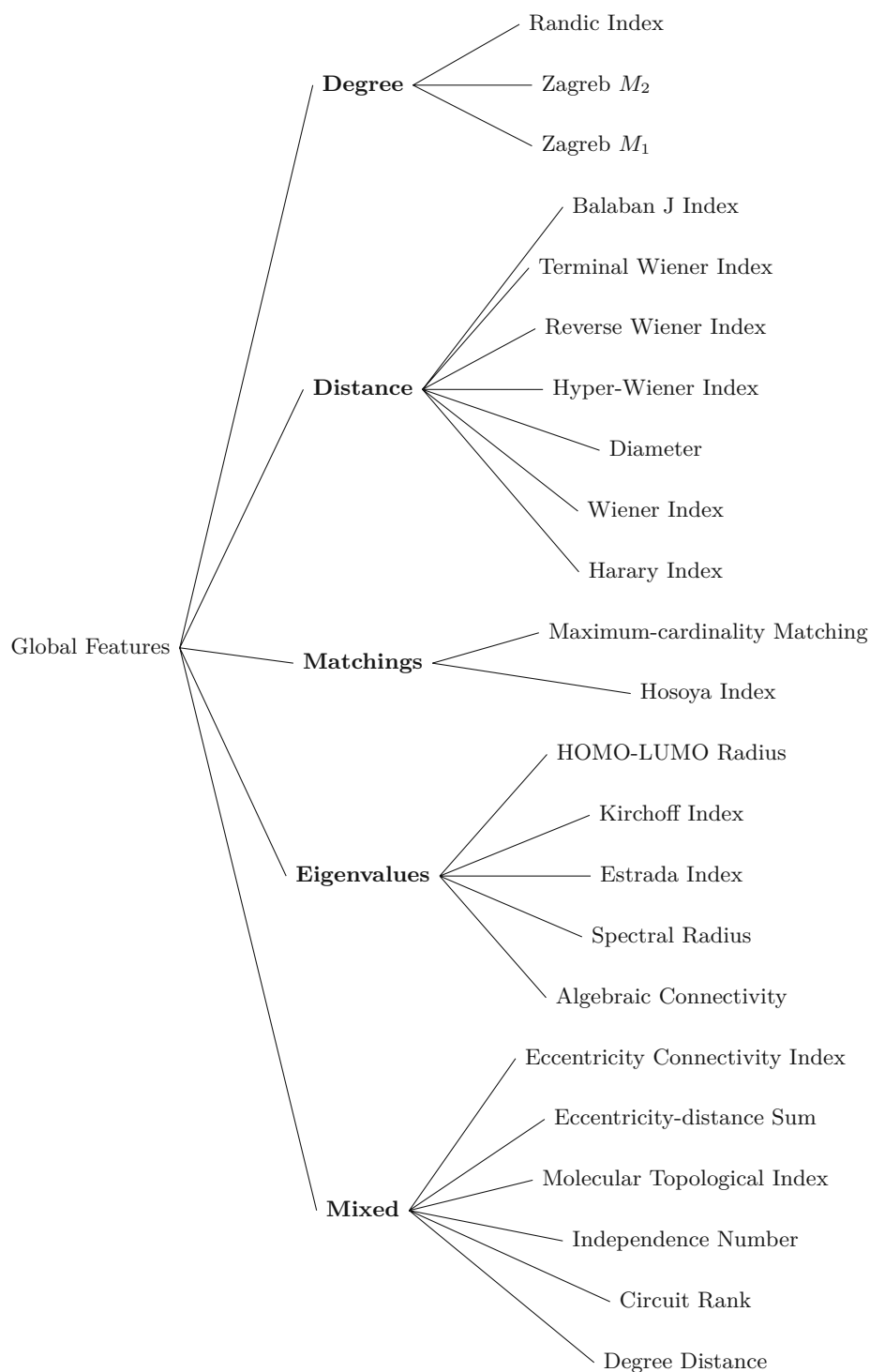


Figure 4.1: Classification of global features based on graph properties such as distance, degree, eigenvalues, matchings, and mixed.

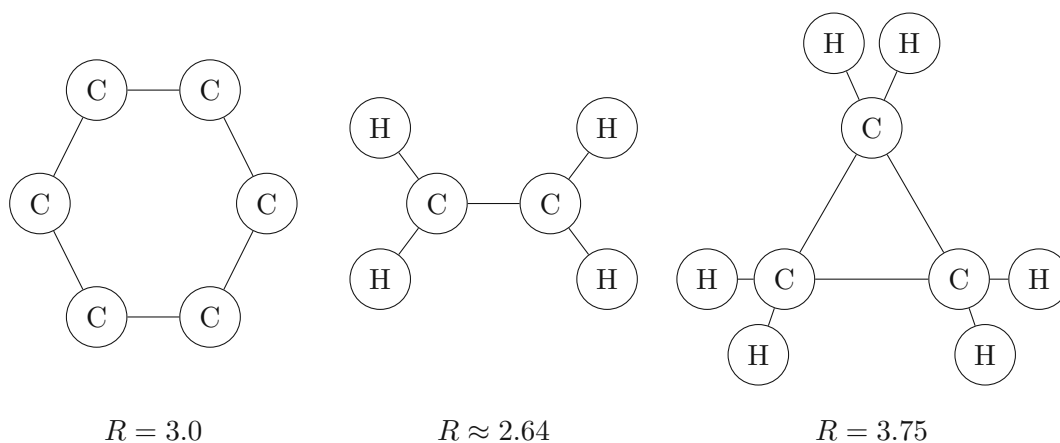


Figure 4.2: Randić index for 3 molecules: benzene, ethylene and cyclopropane.

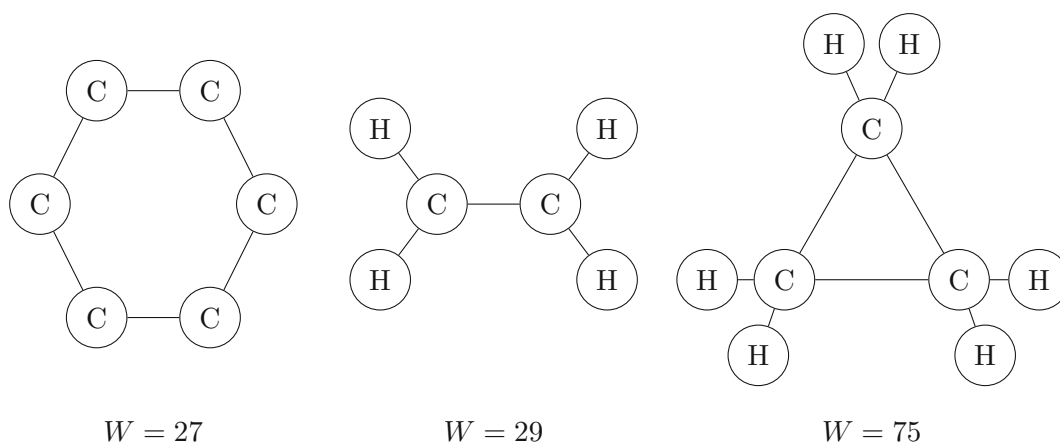


Figure 4.3: Wiener index for 3 molecules: benzene, ethylene and cyclopropane.

- Diameter of a graph [Diestel, 2005][p. 8]:

$$\text{diam}(G) = \max_{u,v \in V(G)} d(u, v).$$

- Hyper-Wiener (WW) index [Randić, 1993]:

$$\text{WW}(G) = \frac{1}{2} \sum_{u,v \in V(G)} (d(u, v) + d^2(u, v)).$$

- Reverse Wiener (Λ) index [Balaban et al., 2000]:

$$\Lambda(G) = \frac{1}{2} n(n-1) \text{diam}(G) - W(G),$$

where n is the number of nodes in graph G .

- Terminal Wiener (TW) index [Gutman et al., 2009; Gutman and Furtula, 2010]:

$$\text{TW}(G) = \sum_{\{u,v\} \subseteq V_p(G)} d(u,v),$$

where $V_p(G)$ is the set of leaf vertices in G .

- Harary index [Plavšić et al., 1993]:

$$H(G) = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n (RD)_{ij},$$

where

$$(RD)_{ij} = \begin{cases} d^{-1}(i,j) & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases}.$$

- Balaban J index [Balaban, 1982]:

$$J(G) = \frac{m}{m-n+2} \sum_{\{u,v\} \in E(G)} \frac{1}{\sqrt{d(u,v)}},$$

where d_i is the sum of all entries in the i -th row (or column) of the graph distance matrix.

4.2.3 Eigenvalue-Based GFs

Eigenvalue-based global features utilize the *eigenvalues* of matrices associated with the graph, such as the adjacency matrix $A = A(G)$ or the Laplacian matrix $L = L(G)$, given a graph G . These indices capture various *spectral properties* of the graph, that reflect its structural characteristics. For example, the *spectral radius* [Gradshteyn et al., 1980] of a graph G is defined as the largest absolute value of the eigenvalues of its adjacency matrix A . If $\lambda_1, \lambda_2, \dots, \lambda_n$ are the eigenvalues of A , then the spectral radius $\rho(G)$ is given by:

$$\rho(G) = \max\{|\lambda_1|, |\lambda_2|, \dots, |\lambda_n|\}.$$

where λ_i are the eigenvalues of the adjacency matrix A of the graph G .

Besides the spectral radius, we select four other GFs that are computed using the eigenvalues of matrices associated with the graph:

- Algebraic connectivity (second smallest eigenvalue of the Laplacian matrix) [Fiedler, 1973]:

$$a(G) = \mu_2(L(G)).$$

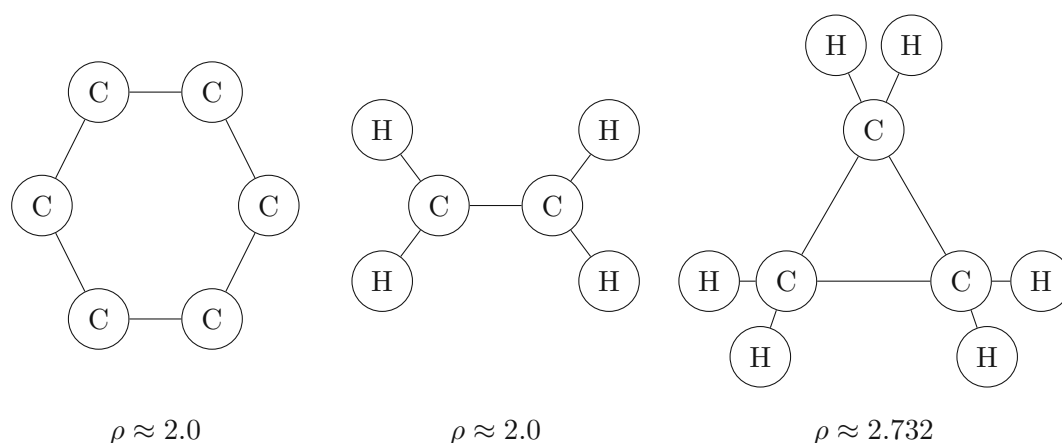


Figure 4.4: Spectral radius for 3 molecules: benzene, ethylene and cyclopropane.

- Estrada (EE) index [Estrada, 2000]:

$$EE(G) = \sum_{j=1}^n e^{\lambda_j},$$

where n is the number of nodes in graph G and $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$ is a non-increasing ordering of the eigenvalues of the adjacency matrix A .

- Kirchoff index [Klein and Randić, 1993; Bonchev et al., 1994]:

$$Kf(G) = n \sum_{k=1}^{n-1} \frac{1}{\mu_k},$$

where $\mu_1 \geq \mu_2 \geq \dots \geq \mu_n = 0$ are the eigenvalues of the Laplacian matrix.

- HOMO-LUMO radius (*HL*-index) [Jaklič et al., 2012]:

$$R_{HL}(G) = \max\{|\lambda_H|, |\lambda_L|\},$$

where $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$ are the eigenvalues of A and $H = \lfloor \frac{n+1}{2} \rfloor$ and $L = \lceil \frac{n+1}{2} \rceil$.

4.2.4 Matching-based GFs

Matching-based global features are computed based on *matchings* in the graph, where a matching is a set of edges without common vertices. These features help in understanding the pairing possibilities within a graph. Such an example is the *Hosoya index*, (also known as the *Z-index*) of a graph G , which has been reported to have a good correlation with the boiling points of alkane isomers. It is defined as the total number of matchings in the graph, including the empty matching [Hosoya, 1971]. Formally, if M_k is the number of k -matchings in G , the Hosoya index $Z(G)$ is given by:

$$Z(G) = \sum_{k=0}^{\lfloor n/2 \rfloor} M_k,$$

where n is the number of vertices in the graph G and $M_0 = 1$ by definition (the empty matching).

It can be computed as well as the sum of all coefficients of the *matching polynomial* when $x = 1$. The matching polynomial $M(G, x)$ of a graph G is given by:

$$M(G, x) = \sum_{k=0}^{\lfloor n/2 \rfloor} \mu(G, k) x^k,$$

where $\mu(G, k)$ is the number of k -matchings in graph G .

Then:

$$Z(G) = M(G, 1) = \sum_{k=0}^{\lfloor n/2 \rfloor} \mu(G, k),$$

where n is the number of vertices in the graph G .

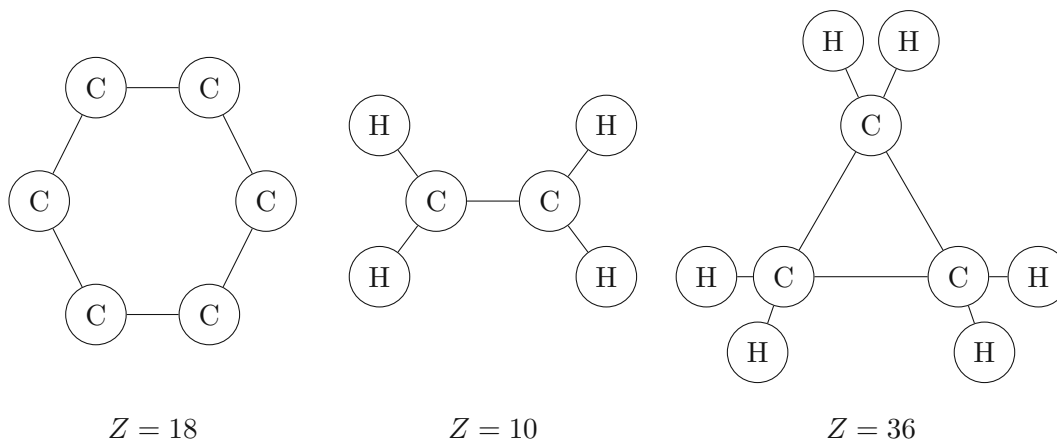


Figure 4.5: Hosoya index for 3 molecules: benzene, ethylene and cyclopropane.

Besides the Hosoya index, we select one other GF that is based on matchings in the graph:

- Size of the maximum-cardinality matching [Manacher, 1989]:

$$\text{MCM}(G) = |\nu(G)|,$$

where $\nu(G)$ is a maximum-cardinality matching in graph G .

4.2.5 Mixed GFs

Mixed global features combine elements from degree, distance, eigenvalue, and matching-based indices, meaning one could use both the distance and degree information within a graph. One such example is the *Degree Distance (DD) index* [Dobrynin and Kochetova, 1994], defined as it follows:

$$DD(G) = \sum_{\substack{u,v \in V(G) \\ u \neq v}} (\deg(u) + \deg(v)) \cdot d(u, v),$$

where m is the number of edges and n is the number of nodes. The summation is taken over all pairs of adjacent vertices.

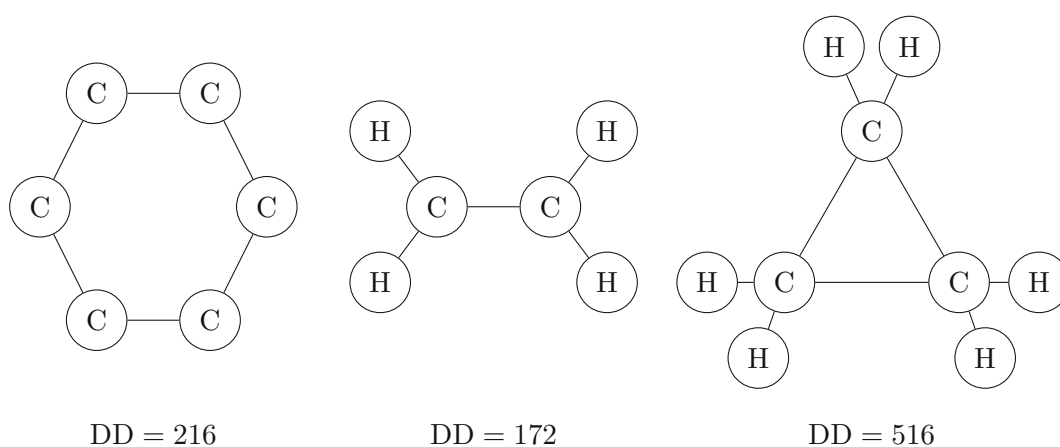


Figure 4.6: Degree Distance index for 3 molecules: benzene, ethylene and cyclopropane.

Besides the Degree Distance (DD) index, we select five other GFs that are computed using mixed properties of the graph:

- Circuit rank [Kirchhoff, 1847]:

$$\mu = m - n + c,$$

where m is the number of edges, n is the number of nodes and c is the number of connected components in the graph.

- Eccentricity-distance sum (EDS) index [Gupta et al., 2002]:

$$\xi^{ds}(G) = \sum_{\{u,v\} \subseteq V(G)} (\varepsilon(u) + \varepsilon(v)) d(u, v),$$

where $\varepsilon(v) = \max_{u \in V(G)} d(u, v)$ is the *eccentricity* of vertex v .

- Eccentricity Connectivity index [Sharma et al., 1997]:

$$\xi(G) = \sum_{u \in V(G)} \deg(u) \cdot \varepsilon(u).$$

- Independence number [Godsil and Royle, 2001][p. 3]:

$$\alpha(G) = |S_{\max}(G)|,$$

where $S_{\max}(G)$ is a maximum *independent* set, i.e., a set of vertices of maximum possible size in G such that no two of which are adjacent.

- Molecular Topological Index (MTI) [Schultz, 1989]:

$$\text{MTI} = \sum_{i=1}^n E_i,$$

where E_i are the components of the vector $E = (A + d)D$.

4.3 Expressivity of GFs

In this section, we analyze the expressivity of various global features (GFs) in relation to the Weisfeiler-Lehman (WL) graph isomorphism test.

Definition 3. A global feature f is *expressivity-increasing* if for a pair of WL algorithm indistinguishable graphs G and H , i.e., $\text{WL}(G) = \text{WL}(H)$, it holds that $f(G) \neq f(H)$.

Expressivity of Global Features. The concatenation of a global graph feature f and a learned graph embedding GNN will never decrease the expressivity of GNN or f . For WL algorithm there exist GFs that can distinguish a pair of graphs which WL alone cannot (more details in Section 4.3).

Definition 4. Let G be a graph, and let $\text{WL}(G)$ denote the output of the WL algorithm for graph G . The combined representation $(\text{WL}, f)(G)$ is then defined as:

$$(\text{WL}, f)(G) = (\text{WL}(G), f(G)).$$

Proposition 1. Let f be an *expressivity-increasing* global graph feature. Then, (WL, f) is strictly more expressive than WL algorithm.

Proof. By the definition of an *expressivity-increasing* global graph feature, we know that for a pair of graphs G and H that are indistinguishable by the WL algorithm, i.e., $\text{WL}(G) = \text{WL}(H)$, it holds that $f(G) \neq f(H)$.

Consider the augmented function (WL, f) , which combines the result of the WL algorithm with the global feature f . For any pair of graphs G and H :

- If $WL(G) \neq WL(H)$, then $(WL, f)(G) \neq (WL, f)(H)$ because the WL algorithm alone distinguishes G and H .
- If $WL(G) = WL(H)$, but $f(G) \neq f(H)$, then $(WL, f)(G) \neq (WL, f)(H)$ because the global feature f distinguishes G and H .

Therefore, (WL, f) distinguishes any pair of graphs that the WL algorithm can distinguish and at least one pair of graphs that the WL algorithm cannot distinguish. This implies that (WL, f) is strictly more expressive than the WL algorithm alone.

□

Next, to understand the distinguishing power of different global features, we evaluate each selected feature to determine whether it satisfies the expressivity-increasing criterion. The purpose of this analysis is to provide insights into which global features can compensate for the previously mentioned limitations of the WL algorithm.

4.3.1 Degree-based GFs expressivity

Lemma 1 (Arvind et al. 2020). *If two graphs are indistinguishable by WL color refinement, they have the same degree sequence.*

Since all of the global features that are computed based on node degrees that we selected depend only on the node sequence, they can be written as a function $f_{ND} : \mathcal{G} \rightarrow \mathbb{R}$, $f_{ND}(G) = g(\text{degseq}(G))$, where $\text{degseq} : \mathcal{G} \rightarrow \mathbb{Z}_{\geq 0}^n$, the degree *sequence* of a graph G , is the multiset² of the degrees of the vertices,

$$\text{degseq}(G) = \{\{\deg(v) : v \in V(G)\}\}.$$

Corollary 1. *Any global graph feature that can be expressed as a function of the degree sequence of a graph is not expressivity-increasing.*

Proof. Suppose that there exist two graphs G_1 and G_2 with $f_{ND}(G_1) \neq f_{ND}(G_2)$, but $WL(G_1) = WL(G_2)$. Hence, by Lemma 1, $\text{degseq}(G_1) = \text{degseq}(G_2)$.

Given that the function $f_{ND}(G)$ depends only on the degree sequence of the graph, we can express it as $f_{ND}(G_1) = g(\text{degseq}(G_1))$ and $f_{ND}(G_2) = g(\text{degseq}(G_2))$.

However, since we have $\text{degseq}(G_1) = \text{degseq}(G_2)$, it follows that $g(\text{degseq}(G_1)) = g(\text{degseq}(G_2))$, and therefore $f_{ND}(G_1) = f_{ND}(G_2)$.

²Usually, the degree sequence of a graph is a list of the degrees, often in descending numerical order. For our purposes here, however, the multiset formulation is more useful, as it implies permutation invariance of the node degrees.

This contradicts our initial assumption that $f_{ND}(G_1) \neq f_{ND}(G_2)$. Therefore, the original statement holds: the function $f_{ND}(G)$ that depends only on the degree sequence of a graph cannot have different values for two graphs with the same node degree sequence.

□

Thus, we conclude that the Randic index, Zagreb M_1 index and Zagreb M_2 index are not expressivity-increasing global features in this context, as they cannot differentiate between two graphs that the WL algorithm cannot distinguish.

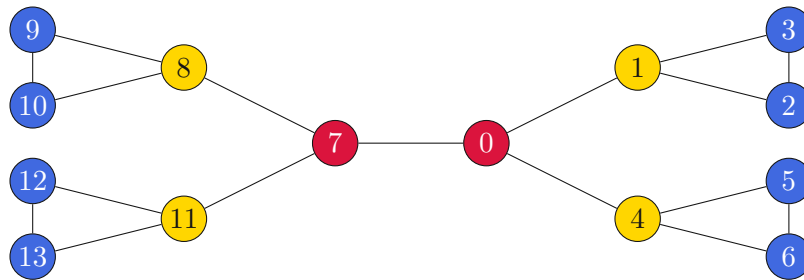


Figure 4.7: G_1

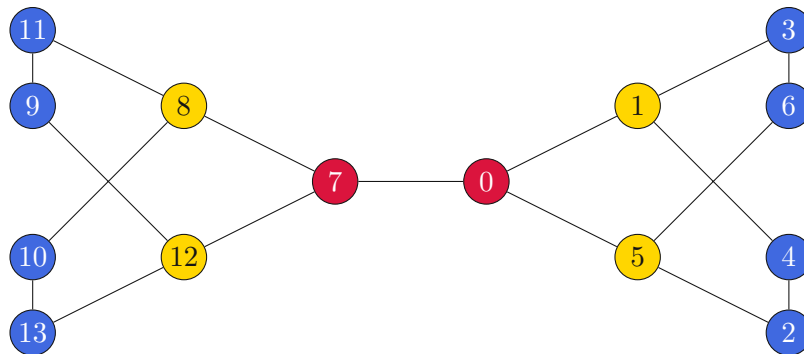


Figure 4.8: G_2

4.3.2 Distance-based GFs expressivity

Consider the two graphs G_1 and G_2 depicted in Figure 4.7 and Figure 4.8. We compute six distance-based GFs for both G_1 and G_2 : Wiener index, Hyper-Wiener index, Reverse Wiener index, Terminal Wiener index, Harary index and Balaban J index. We observe that they yield different values (see Table 4.1).

Proposition 2. *Wiener index, Hyper-Wiener index, Reverse Wiener index, Terminal Wiener index, Harary index and Balaban J index are expressivity-increasing global features.*

Table 4.1: Comparison of distance-based global features values for graphs G_1 and G_2 .

Global Feature	G_1	G_2
Wiener (W)	281	261
Hyper-Wiener (WW)	659	591
Reverse Wiener (Λ)	174	194
Terminal Wiener (TW)	116	94
Harary (H)	39.86	42.53
Balaban (J)	1.541	1.612

Proof. Let G_1, G_2 be the graphs in Figure 4.7 and Figure 4.8 respectively. It can be easily verified that $WL(G_1) = WL(G_2)$ (the stable coloring is shown in Figure 4.7 and Figure 4.8).

As $f(G_1) \neq f(G_2)$, for all $f \in \{W, WW, \Lambda, TW, H, J\}$ (see Table 4.1), by Definition 3, it holds that f is expressivity-increasing. □

Thus, we conclude that the Wiener index, Harary index, Balaban J index, Hyper-Wiener index, Reverse Wiener index and Terminal Wiener index are expressivity-increasing global features in this context, as they can differentiate between two graphs that the WL algorithm cannot distinguish.

Now let's consider the third pair of graphs, G_5 and G_6 , shown below in Figure 4.11 and Figure 4.12. We compute the diameter of a graph for both G_5 and G_6 and observe that it yields different results (see Table 4.2).

Table 4.2: Comparison of diameter values for graphs G_5 and G_6 .

Global Feature	G_5	G_6
Diameter (diam)	3	4

Proposition 3. *The diameter of a graph is an expressivity-increasing global feature.*

Proof. Let G_5, G_6 be the graphs in Figure 4.11 and Figure 4.12 respectively. It can be verified that $WL(G_5) = WL(G_6)$.

As $f(G_5) \neq f(G_6)$, for all $f \in \{\text{diam}\}$ (see Table 4.2), by Definition 3, it holds that f is expressivity-increasing. □

Thus, we conclude that the diameter of a graph is an expressivity-increasing global feature in this context, as it can differentiate between two graphs that the WL algorithm cannot distinguish.

4.3.3 Matching-based GFs expressivity

Consider the two graphs G_1 and G_2 depicted in Figure 4.7 and Figure 4.8. We compute two matching-based GFs for both G_1 and G_2 : Hosoya index and Maximum-cardinality Matching. We observe that they yield different values (see Table 4.3).

Table 4.3: Comparison of matching-based global features values for graphs G_1 and G_2 .

Global Feature	G_1	G_2
Hosoya (Z)	1280	1480
Maximum-cardinality Matching (MCM)	6	7

Proposition 4. *Hosoya index and Maximum-cardinality Matching are expressivity-increasing global features.*

Proof. Let G_1, G_2 be the graphs in Figure 4.7 and Figure 4.8 respectively. It can be verified that $WL(G_1) = WL(G_2)$ (the stable coloring is shown in Figure 4.7 and Figure 4.8).

As $f(G_1) \neq f(G_2)$, for all $f \in \{Z, \text{MCM}\}$ (see Table 4.3), by Definition 3, it holds that f is expressivity-increasing. \square

Thus, we conclude that the Hosoya index and Maximum-cardinality Matching are expressivity-increasing global features in this context, as they can differentiate between two graphs that the WL algorithm cannot distinguish.

4.3.4 Eigenvalue-based GFs expressivity

Lemma 2 (Arvind et al. 2020). *If two graphs are indistinguishable by WL color refinement, they have the same maximum eigenvalue.*

Corollary 2. *The maximum eigenvalue of the adjacency matrix A as a global feature is not expressivity-increasing.*

Proof. Suppose that there exist two graphs G_1 and G_2 with $\rho(G_1) \neq \rho(G_2)$, but $WL(G_1) = WL(G_2)$. Hence, by Lemma 2, $\lambda_{\max}(G_1) = \lambda_{\max}(G_2)$.

As $A_{ij} \geq 0$, for all $i, j \in V$ by the *Perron-Frobenius* theorem, there is a positive real number r such that r is an eigenvalue of A and any other eigenvalue λ in absolute value is strictly smaller than r , $|\lambda| \leq r$. Thus, $\rho(G) = r$, and hence $r = \lambda_{\max}$.

Since we have $\lambda_{\max}(G_1) = \lambda_{\max}(G_2)$, it follows that $\rho(G_1) = \rho(G_2)$.

This contradicts our initial assumption that $\rho(G_1) \neq \rho(G_2)$. Therefore, the original statement holds: two graphs that have equal maximum eigenvalues will have the same maximum absolute values of the eigenvalues of A . \square

Thus, we conclude that the spectral radius is not an expressivity-increasing global feature in this context, as it cannot differentiate between two graphs that the WL algorithm cannot distinguish.

Expressivity-increasing features. Consider the two graphs G_3 and G_4 depicted in Figure 4.9 and Figure 4.10. We compute four eigenvalue-based GFs for both G_3 and G_4 : HOMO-LUMO index, Kirchoff index, Estrada index and Algebraic Connectivity. We observe that they yield different values (see Table 4.4).

Table 4.4: Comparison of eigenvalue-based global features values for graphs G_3 and G_4 .

Global Feature	G_3	G_4
HOMO-LUMO (R_{HL})	2	1
Kirchoff (Kf)	15.515	43.578
Estrada (EE)	13.696	16.249
Algebraic Connectivity (a)	1	0

Proposition 5. *HOMO-LUMO index, Kirchoff index, Estrada index and the Algebraic Connectivity are expressivity-increasing global features.*

Proof. Let G_3, G_4 be the graphs in Figure 4.9 and Figure 4.10 respectively. It can be verified that $WL(G_3) = WL(G_4)$.

As $f(G_3) \neq f(G_4)$, for all $f \in \{R_{HL}, Kf, EE, a\}$ (see Table 4.4), by Definition 3, it holds that f is expressivity-increasing. \square

Thus, we conclude that the HOMO-LUMO index, Kirchoff index, Estrada index and the Algebraic Connectivity are expressivity-increasing global features in this context, as they can differentiate between two graphs that the WL algorithm cannot distinguish.

4.3.5 Mixed GFs expressivity

Consider the two graphs G_1 and G_2 depicted in Figure 4.7 and Figure 4.8. We compute four mixed GFs for both G_1 and G_2 : Independence Number, Molecular Topological Index, Eccentricity-distance Sum and Degree Distance. We observe that they yield different values (see Table 4.5).

Proposition 6. *Independence Number, Molecular Topological Index, Eccentricity-distance Sum and Degree Distance are expressivity-increasing global features.*

Proof. Let G_1, G_2 be the graphs in Figure 4.7 and Figure 4.8 respectively. It can be verified that $WL(G_1) = WL(G_2)$ (the stable coloring is shown in Figure 4.7 and Figure 4.8).

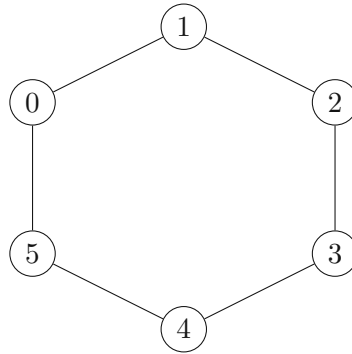


Figure 4.9: G_3



Figure 4.10: G_4

Table 4.5: Comparison of mixed global features values for graphs G_1 and G_2 .

Global Feature	G_1	G_2
Independence Number (α)	5	6
Molecular Topological Index (MTI)	1404	1314
Eccentricity-distance Sum (ξ^{ds})	2562	2370
Degree Distance (DD)	2636	2496

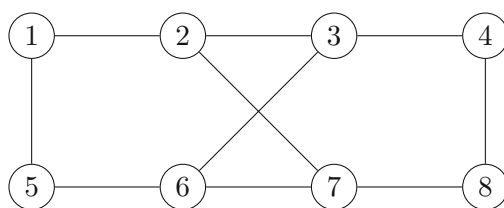
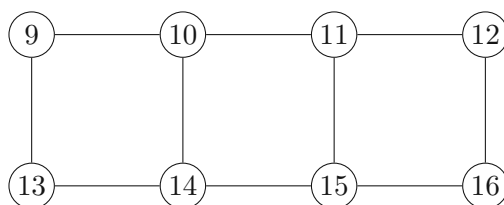
As $f(G_1) \neq f(G_2)$, for all $f \in \{\alpha, \text{MTI}, \xi^{ds}, \text{DD}\}$ (see Table 4.5), by Definition 3, it holds that f is expressivity-increasing. \square

Thus, we conclude that the Independence Number, Molecular Topological Index, Eccentricity-distance Sum and Degree Distance are expressivity-increasing global features in this context, as they can differentiate between two graphs that the WL algorithm cannot distinguish.

Consider now the two graphs, G_3 and G_4 , depicted in Figure 4.9 and Figure 4.10. We compute two mixed GFs for both G_3 and G_4 : Circuit Rank and the Eccentricity Connectivity Index. We observe that they yield different values (see Table 4.6).

Table 4.6: Comparison of mixed global features values for graphs G_3 and G_4 .

Global Feature	G_3	G_4
Circuit Rank (μ)	1	2
Eccentricity Connectivity (ξ)	36	∞

Figure 4.11: G_5 Figure 4.12: G_6

Proposition 7. *Circuit Rank and the Eccentricity Connectivity index are expressivity-increasing global features.*

Proof. Let G_3, G_4 be the graphs in Figure 4.9 and Figure 4.10 respectively. It can be verified that $WL(G_3) = WL(G_4)$.

As $f(G_3) \neq f(G_4)$, for all $f \in \{\mu, \xi\}$ (see Table 4.6), by Definition 3, it holds that f is expressivity-increasing. \square

Thus, we conclude that the Circuit Rank and the Eccentricity Connectivity index are expressivity-increasing global features in this context, as they can differentiate between two graphs that the WL algorithm cannot distinguish.

A detailed summary of the selected global features is presented in Table 4.7. We highlight two important results:

- Hosoya (Z) and Independence Number (α) as they are *NP-hard*. However, on small molecular graphs they can still be computed efficiently, e.g., using the matching polynomial.
- The global features identified as not expressivity-increasing, such as Randic (R), Zagreb M_1 , Zagreb M_2 , and Spectral Radius (ρ), indicate that, while computationally feasible, do not contribute to the distinguishing power of the WL algorithm. Nonetheless, they may still prove to be valuable empirically. The degree-based global features have lower computational complexity, with time complexities ranging from $O(n)$ to $O(m)$, yet they may not capture as much structural information as other global features.

- Circuit rank only helps with distinguishing disconnected graphs. On connected graphs, it is not expressivity increasing.

Table 4.7: Summary of the global features expressiveness and runtime complexity.

Global Feature	Expressivity-Increasing	Complexity
Randic (R)	No	$O(m)$
Zagreb M_1	No	$O(n)$
Zagreb M_2	No	$O(m)$
Balaban (J)	Yes	$O(n^3)$
Terminal Wiener (TW)	Yes	$O(n^2)$
Reverse Wiener (Λ)	Yes	$O(n^2)$
Hyper-Wiener (WW)	Yes	$O(n^2)$
Diameter (diam)	Yes	$O(nm)$
Wiener (W)	Yes	$O(n^2)$
Harary (H)	Yes	$O(n^3)$
Max. card. Matching (MCM)	Yes	$O(nm)$
Hosoya (Z)	Yes	NP-hard
HOMO-LUMO (R_{HL})	Yes	$O(n^2)$
Kirchhoff (Kf)	Yes	$O(n^3)$
Estrada (EE)	Yes	$O(n^3)$
Spectral Radius (ρ)	No	$O(n^3)$
Algebraic Connectivity (a)	Yes	$O(n^3)$
Eccentricity Connectivity (ξ)	Yes	$O(nm)$
Eccentricity-distance Sum (ξ^{ds})	Yes	$O(nm)$
Molecular Topological Index (MTI)	Yes	$O(n^2)$
Independence Number (α)	Yes	NP-hard
Circuit Rank (μ)	Yes	$O(m)$
Degree Distance (DD)	Yes	$O(n^2)$

4.4 GNNs with Global Features

We propose a method of incorporating global graph features with the generated embeddings of GNNs, so that we provide the GNN with additional context that leverages both local node information and global structural insights. With this integration method, we aim to (1) improve the expressivity of the overall architecture and (2) enhance its predictive performance on graph-level tasks.

An overview of the proposed architecture can be seen in Figure 4.13. We combine the learned graph embedding function of a GNN with another permutation invariant function on graphs, i.e., the global feature, by concatenating their outputs.

For MPNNs this implies that we can increase their expressivity with global graph features.

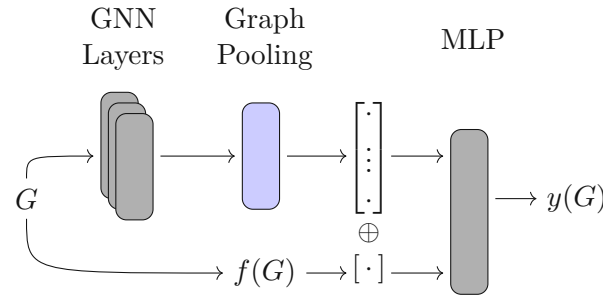


Figure 4.13: Our proposed GNN architecture. We concatenate global graph features to the output of the graph pooling layer after multiple GNN layers. Figure adapted from Welke et al. [2023].

Theorem 2. *Let f be a message passing neural network (MPNN) and let g be a global feature. If the output of the MPNN, $f(G)$, is concatenated with the global feature $g(G)$ and the resulting vector is passed through a multilayer perceptron (MLP) that computes an injective function, then the combined function*

$$h(G) = \text{MLP}([f(G), g(G)])$$

is strictly more expressive than the original MPNN f without the concatenated global feature.

Proof. Let f be a message passing neural network (MPNN) that computes a function $f : \mathcal{G} \rightarrow \mathbb{R}^d$, where \mathcal{G} is the set of all graphs and \mathbb{R}^d is a d -dimensional real vector space representing the output of the MPNN for a graph $G \in \mathcal{G}$. Let $g : \mathcal{G} \rightarrow \mathbb{R}$ be a global feature that maps each graph G to a one-dimensional real vector.

We define a new function $h : \mathcal{G} \rightarrow \mathbb{R}^m$ as follows:

$$h(G) = \text{MLP}([f(G), g(G)])$$

where $[f(G), g(G)] \in \mathbb{R}^{d+1}$ is the concatenation of the MPNN output $f(G)$ and the global feature $g(G)$, and $\text{MLP} : \mathbb{R}^{d+1} \rightarrow \mathbb{R}^m$ is a multilayer perceptron that computes an injective function.

Assume $f(G_1) = f(G_2)$ but $g(G_1) \neq g(G_2)$:

By assumption, the MPNN f is not expressive enough to distinguish between graphs G_1 and G_2 . Therefore, we have:

$$f(G_1) = f(G_2).$$

However, the global features of G_1 and G_2 differ:

$$g(G_1) \neq g(G_2).$$

Consider the concatenated vectors:

$$[f(G_1), g(G_1)] \quad \text{and} \quad [f(G_2), g(G_2)].$$

Since $f(G_1) = f(G_2)$ but $g(G_1) \neq g(G_2)$, we have:

$$[f(G_1), g(G_1)] \neq [f(G_2), g(G_2)].$$

This follows because the concatenation operation preserves the distinction between different components, i.e., if two vectors are not identical, their concatenation will also not be identical.

Since $\text{MLP} : \mathbb{R}^{d+1} \rightarrow \mathbb{R}^m$ is an injective function, it preserves distinctness. Therefore, if:

$$[f(G_1), g(G_1)] \neq [f(G_2), g(G_2)],$$

then:

$$\text{MLP}([f(G_1), g(G_1)]) \neq \text{MLP}([f(G_2), g(G_2)]).$$

Thus, we have:

$$h(G_1) \neq h(G_2).$$

We have shown that for graphs G_1 and G_2 , if $f(G_1) = f(G_2)$ but $g(G_1) \neq g(G_2)$, then $h(G_1) \neq h(G_2)$. Therefore, the function $h(G)$ is more expressive than the original MPNN function $f(G)$.

Since the injective MLP can distinguish between any pair of different concatenated vectors, the combined function $h(G)$ can capture more information about the graph than the original MPNN, making it strictly more expressive.

□

Experiments

In this chapter, we investigate the performance of GNNs with Global Features (GFs) attached on both graph regression and classification tasks. The experiments¹ are designed to determine whether attaching GFs with the base GNNs will improve the predictive performance over just the GNN alone. Furthermore, we investigate whether GF significantly improves the empirical performance of graph-level tasks.

5.1 Experimental Setup

Models. We compare the empirical performance of three graph neural networks (GNNs): the Graph Isomorphism Network (GIN) as described by Xu et al. [2019] (more details are provided in Section 2.5), the Graph Convolutional Network (GCN) as introduced by Kipf and Welling [2017], and the CW Network (CWN) as presented by Bodnar et al. [2021]. We evaluate these models against their corresponding global feature (GF) variants, where global features are attached to the embeddings generated by the GNNs.

The GCN applies convolutional operations to graph data, effectively capturing local neighborhood information within the graph. The CWN, on the other hand, incorporates cellular lifting maps to perform message passing on cell complexes. This approach enables the model to capture higher-order topological features of the graph, which are not accessible through traditional GNNs. For the CWN, we use a setup similar to the one described by Jogl et al. [2022], i.e., we simulate it by applying a graph transformation and then run it through an MPNN (in this case, GIN).

We tune hyperparameters for the GNN baselines with no global graph features attached. We train the GNN until the stopping criteria is met, and then attach the GFs to the graphs, load the pre-trained model (baseline) and further do fine-tuning of the MLP

¹Code for reproducing the experiments can be found at <https://github.com/andreibrasoveanu97/gnn-global-features>

layers. The hyperparameter configurations used in our experiments are presented in Table 5.1 and Table 5.2. The column containing the hyperparameters remains constant across different datasets, even though not all hyperparameters are applicable to every dataset. For example, in the case of the `ogbg-molhiv` dataset, we did not use a learning rate *scheduler* or *decay*.

We also introduce two additional experiments, that are using the same set of hyperparameters corresponding to their method, as shown in the tables below. We therefore experiment with a setup of:

- A constant “1”, i.e., a global feature $f(G) = 1$, for all $G \in \mathcal{G}$, for an ablation study with the purpose of investigating if any practical performance increases are due to additional finetuning and additional weights in the MLP.
- An experiment combining all of the global features for a dataset to evaluate if their collective use yields better performance compared to using each feature individually.

Node-broadcasted GFs. In Chapter 4 we described how attaching the global features should be done at the graph-level, as they are invariant to it. However, to further validate this hypothesis, we empirically evaluate a node-broadcasted variant of our proposed architecture, denoted GF_{NB} . In this case, we attach the global feature to each node, adding an additional dimension to the node feature vector. We evaluate this setup with two different GNNs: CWN (simulated by a GIN using graph transformations) and GCN.

Datasets. We evaluate our models on the graph classification dataset `ogbg-molhiv` [Hu et al., 2020] and on a graph regression dataset, ZINC [Sterling and Irwin, 2015; Gómez-Bombarelli et al., 2018]. The `ogbg-molhiv` dataset contains 41,127 graphs and is a binary graph classification problem, where the task is to predict whether a molecule inhibits HIV. For the ZINC dataset, we use the smaller variant that contains 12,000 molecular graphs, following the approach taken by Barceló et al. [2021] and Bouritsas et al. [2023]. The ZINC dataset is a graph regression problem with one regression target, which is predicting the constrained solubility, denoted as $\log P$, of a molecule.

As defined in the Open Graph Benchmark (OGB) [Hu et al., 2020], we use the area under the receiver operating characteristic curve (ROC-AUC) to evaluate the results on `ogbg-molhiv`. For ZINC, we use the mean absolute error (MAE) to evaluate the regression target.

Setup. We use a setup similar to Dwivedi et al. [2024]. For each dataset we evaluate at least one type of GNN combined with a global feature concatenated to the generated embeddings. Both ZINC and `ogbg-molhiv` datasets have preset train, validation and test splits for their data.

For all of the methods, we aim to tune the following hyperparameters: the number of epochs, the dropout probability, the embedding dimension (or hidden layer size for MLP),

the batch size, the pooling method, the number of GNN layers and the number of MLP layers. There are a few exceptions:

- For the CWN+GF variants, we do not tune the batch size hyperparameter because this setup requires a significantly increased amount of time compared to other configurations. For instance, with the ZINC dataset, the grid search already involves more than 1,000 possible hyperparameter combinations. However, we do tune the size of the largest cycle to lift, referred to as `Max Ring Size`.
- For the `ogbg-molhiv` dataset, the training duration is significantly longer compared to the other two datasets; therefore, we set the number of epochs to 100.

During training, we use a learning rate of 10^{-3} combined with the *ReduceLROnPlateau* scheduler. Specifically, whenever the model does not show improvement on the validation set for 20 epochs, we reduce the learning rate by 50 percent. Training is stopped when the learning rate falls below 10^{-5} or after the maximum number of epochs.

For evaluation, we select the metric obtained in the epoch with the best validation performance. We then report the average and standard deviation of this metric over 10 separate training runs, each initialized with a different random seed. The loss function used is the *L1 loss* for the ZINC and QM9 dataset (averaged across the 19 regression targets) and the *binary cross-entropy loss* for the `ogbg-molhiv` dataset.

Global features missing. In Section 5.2, it can be observed that the global feature list is not consistent across the different experiments. There are a few reasons as to why this the case:

- Hosoya index (Z) is absent from the `ogbg-molhiv` dataset as the script that computes the global features for the graphs never finished running (stopped after approx. 96 hours).
- There are global features in our list which rely on the graph being connected, e.g., graph-diameter, eccentricity-based features. For these graphs, the associated value of these global features is ∞ . A possible solution would be to assign them a conventional numerical value that is outside the range of possible values, e.g., -1 for the graph diameter. For the sake of simplicity and to avoid the risk of skewing the distribution of the global feature values, we left these features out for their corresponding datasets.
- Due to time constraints and limited computational resources, a few combinations of GNNs with GFs have been left out.

5.2 Results and Discussion

ZINC. In Table 5.3, Table 5.4 and Table 5.5 we show the experimental results for the ZINC dataset. **Bold** values are better than their corresponding baseline results by at least one standard deviation for a given global feature. **Blue** values are the best result for a given GF variant. **Red** values indicate results which are worse than the baseline value. We analyze the effectiveness of the global features across the different models:

- **CWN+GF:** Most global features contribute to a reduction in MAE compared to the baseline CWN model, indicating that adding global features generally improves model performance. Notable improvements are seen with features like the Wiener Hyper Index (WW, -5.68%), Algebraic Connectivity (a , -4.88%), and Estrada Index (EE, -4.72%). However, some features like Circuit Rank (μ , +0.51%) and Spectral Radius (ρ , +1.15%) even slightly worsen the performance when using the GF_{NB} variant.
- **GCN+GF:** This model also benefits from the inclusion of most global features, with significant improvements from features like the Molecular Topological Index (MTI, -15.84%) and Circuit Rank (μ , -8.91%). However, some features, such as Degree Distance (DD, +9.54%) and Balaban J (J , +4.06%), cause substantial increases in MAE for the GF_{NB} variant. Since the node broadcasting variant has the global feature numerical value concatenated to each node feature vector, results may vary depending on the GNN layer used with this method.
- **GIN+GF:** The GIN model shows smaller but consistent improvements across all global features, with the best results for "All features" (-11.83%) and individual features like Algebraic Connectivity (a , -6.95%) and Maximal Cardinality Matching (MCM, -5.91%). This indicates that while the performance gain from a single global feature is not too large, combining multiple such features can give significant performance improvements without changing the GNN architecture.

The Algebraic Connectivity (a) shows consistent reductions in MAE in all models (CWN: -4.88%, GCN: -4.83%, GIN: -6.95%), suggesting it is a valuable global feature. Interestingly, we do not observe any significant difference in performance across this dataset between the set of expressivity-increasing GFs and the ones which are not expressivity-increasing. The constant 1's global feature seems to be either deteriorating the performance or slightly increasing it, although the improvement is minimal and could be within the margin of error.

ogbg-molhiv. In Table 5.6, Table 5.7 and Table 5.8 we show the experimental results for the ogbg-molhiv dataset. **Bold** values are better than their corresponding baseline results by at least one standard deviation for a given global feature. **Blue** values are the best result for a given GF variant. **Red** values indicate results which are worse than the

baseline value. We analyze the effectiveness of the global features across the different models:

- **CWN+GF**: The integration of global features has a mixed impact on the ROC-AUC for the CWN model. Some features improve performance, such as the Estrada Index (EE, +2.30%), Maximal Cardinality Matching (MCM, +2.24%), and Kirchhoff index (Kf , +2.08%). However, several features also reduce performance, including Algebraic Connectivity (a , -2.19%), Wiener Hyper (WW, -2.36%), and Wiener Terminal (TW, -2.02%).
- **GCN+GF**: GCN generally shows improvement with some global features, such as the Molecular Topological Index (MTI, +3.36%) and Zagreb M_2 (+3.22%). However, several features, particularly in the GF_{NB} variant, negatively impact performance, such as Spectral Radius (ρ , -2.08%) and Wiener Index (W , -2.15%). This indicates that for GCN choosing the integration method of the global features is highly impactful on the evaluation results.
- **GIN+GF**: GIN shows minor changes in ROC-AUC with the addition of global features. The Circuit Rank (μ , +0.70%) and Zagreb M_2 (+0.61%) features show the most notable improvements, while the Independence Number (α , -0.18%) and $f(G) = 1$ (-0.95%) slightly decrease performance. Overall, GIN appears less sensitive to global feature additions than the other models.

Both Harary (H) and Kirchhoff (Kf) indices show consistent positive results in both CWN and GCN models, suggesting they may be valuable global features for this dataset. We observe that for GCN, results are highly dependable on the choice of the integration method for the GFs, with GF_{NB} variant often leading to decreased performance. As we had concluded with the ZINC dataset, the expressivity of the global feature does not have a direct impact on its performance, as we see non-expressivity-increasing features like the Zagreb indices performing competitively to the expressivity-increasing features.

Overall, we see that using GNNs with global features increases the predictive performance for graph-level tasks. We observe that the performance of the global features can be specific to a dataset: Algebraic Connectivity (a) exhibits great performance across all models for the ZINC dataset, but at the same time it performs consistently poor for the `ogbg-molhiv` variants. Interestingly, non-expressivity-increasing features perform on par with the expressivity-increasing global features. One reason could be that there is not a strong need of more expressive models when evaluating them on the unseen data and WL can already distinguish the majority of graphs in many datasets [Zopf, 2022]. The node-broadcasting variant performs worse on average, as we even observe consistent decreases in performance compared to the baseline for the GCN with the `ogbg-molhiv` dataset. Comparing the results across the two datasets, we notice that the `ogbg-molhiv` dataset is higher in variability, which can be a result of the more restrictive training conditions: we use a lower number of epochs (100) and no learning rate decay.

Table 5.1: Hyperparameter grid used for the ZINC dataset.

Hyperparameters	CWN + GF	GIN + GF	GCN + GF
Epochs	500, 1000	500, 1000	500, 1000
Batch size	128	32, 64, 128	32, 64, 128
Drop rate	0.0, 0.5	0, 0.5	0, 0.5
Embedding dim.	64, 128, 256	64, 128, 256	128, 256, 300
Pooling method	sum, mean	sum, mean	sum, mean
Learning rate	1e-3	1e-3	1e-3
Minimum LR	1e-5	1e-5	1e-5
LR decay rate	0.5	0.5	0.5
LR decay steps	20	20	20
LR scheduler	ReduceOnPlateau	ReduceOnPlateau	ReduceOnPlateau
Max Ring Size	6, 8, 10	N/A	N/A
Number of layers	1, 2, 3, 4, 5	1, 2, 3, 4, 5	1, 2, 3, 4, 5
Number of MLP layers	1, 2, 3	1, 2, 3	1, 2, 3

Table 5.2: Hyperparameter grid used for the ogbg-molhiv dataset.

Hyperparameters	CWN + GF	GIN + GF	GCN + GF
Epochs	100	100	100
Batch size	32	32	32
Drop rate	0.0, 0.5	0, 0.5	0, 0.5
Embedding dim.	64, 128, 256	64, 128, 256	128, 256, 300
Pooling method	mean	mean	mean
Learning rate	1e-3	1e-3	1e-3
Minimum LR	N/A	N/A	N/A
LR decay rate	N/A	N/A	N/A
LR decay steps	N/A	N/A	N/A
LR scheduler	N/A	N/A	N/A
Max Ring Size	6, 8, 10	N/A	N/A
Number of layers	1, 2, 3, 4, 5	1, 2, 3, 4, 5	1, 2, 3, 4, 5
Number of MLP layers	1, 2, 3	1, 2, 3	1, 2, 3

Table 5.3: Relative change in mean absolute error (MAE ↓) on ZINC dataset, CWN+GF variants, over their respective (global-feature free) baselines (lower is better). The baseline CWN achieves 0.0785 ± 0.003 .

Global feature f	CWN + GF	CWN + GF _{NB}
Algebraic Connectivity (a)	-4.88%	-3.15%
Balaban (J)	-3.06%	-2.68%
Circuit Rank (μ)	-3.95%	+0.51%
Degree Distance (DD)	-3.75%	-4.76%
Diameter (diam)	-2.93%	-3.06%
Eccentric Connectivity (ξ)	-4.32%	-3.33%
Eccentricity-distance Sum (ξ^{ds})	-4.31%	-4.31%
Estrada (EE)	-4.72%	-3.54%
Harary index (H)	-2.84%	-3.76%
HOMO-LUMO (R_{HL})	-4.69%	-2.53%
Hosoya (Z)	-4.22%	-1.90%
Independence Number (α)	-4.34%	-1.91%
Kirchhoff index (Kf)	-3.63%	-4.11%
Max. card. Matching (MCM)	-3.58%	+1.06%
Molecular Topological Index (MTI)	-2.69%	-6.27%
Randic (R)	-3.46%	-3.30%
Spectral Radius (ρ)	-3.42%	+1.15%
Wiener (W)	-3.07%	-3.43%
Wiener Hyper (WW)	-5.68%	-3.42%
Wiener Reverse (Λ)	-3.07%	+1.16%
Wiener Terminal (TW)	-4.10%	-2.91%
Zagreb M_1	-3.70%	-2.94%
Zagreb M_2	-2.82%	-2.94%

Table 5.4: Relative change in mean absolute error (MAE ↓) on ZINC dataset, GCN+GF variants, over their respective (global-feature free) baselines (lower is better). The baseline GCN achieves 0.1951 ± 0.009 .

Global feature f	GCN + GF	GCN + GF _{NB}
Algebraic Connectivity (a)	-4.83%	-3.05%
Balaban (J)	-5.55%	+4.06%
Circuit Rank (μ)	-8.91%	-3.75%
Degree Distance (DD)	-3.54%	+9.54%
Diameter (diam)	-4.36%	-5.07%
Eccentric (e)	-2.47%	-2.44%
Eccentric Connectivity (ξ)	-4.32%	-3.33%
Eccentricity-distance Sum (ξ^{ds})	-6.50%	-2.39%
Estrada (EE)	-2.96%	-4.12%
Harary index (H)	-4.75%	-4.87%
HOMO-LUMO (R_{HL})	-4.72%	-0.88%
Hosoya (Z)	-0.93%	-7.15%
Kirchhoff index (Kf)	+0.89%	-5.82%
Max. card. Matching (MCM)	-3.76%	-4.55%
Molecular Topological Index (MTI)	-15.84%	-4.12%
Randic (R)	-2.53%	-5.55%
Spectral Radius (ρ)	-0.49%	-0.52%
Wiener (W)	-7.29%	-2.44%
Wiener Hyper (WW)	-4.24%	-4.42%
Wiener Reverse (Λ)	-3.52%	-3.71%
Wiener Terminal (TW)	-3.07%	-2.12%
Zagreb M_1	-4.08%	-1.26%
Zagreb M_2	-2.31%	-1.47%
All features	-8.54%	-4.26%
$f(G) = 1$	+2.33%	-0.4%

Table 5.5: Relative change in mean absolute error (MAE \downarrow) on ZINC dataset, GIN+GF variant, over its respective (global-feature free) baseline (lower is better). The baseline GIN achieves 0.1854 ± 0.004 .

Global feature f	GIN + GF
Algebraic Connectivity (a)	-6.95%
Circuit Rank (μ)	-1.65%
Hosoya (Z)	-2.17%
Max. card. Matching (MCM)	-5.91%
Spectral Radius (ρ)	-1.63%
Wiener (W)	-5.35%
Zagreb M_1	-5.85%
Zagreb M_2	-3.28%
All features	-11.83%
$f(G) = 1$	-1.4%

Table 5.6: Relative change in area under the receiver operating characteristic curve (ROC-AUC \uparrow) on ogbg-molhiv dataset, CWN+GF variants, over their respective (global-feature free) baselines (higher is better). The baseline CWN achieves 0.7810 ± 0.014 .

Global feature f	CWN + GF	CWN + GF _{NB}
Algebraic Connectivity (a)	-2.19%	+0.41%
Circuit Rank (μ)	+1.61%	+1.14%
Eccentric Connectivity (e)	-1.06%	+0.29%
Eccentricity-distance Sum (ξ^{ds})	-1.27%	+0.09%
Estrada (EE)	+2.30%	+1.13%
Harary index (H)	+1.97%	+1.05%
HOMO-LUMO (R_{HL})	+2.08%	+0.68%
Independence Number (α)	+1.69%	-0.45%
Kirchhoff index (Kf)	+2.08%	+0.68%
Max. card. Matching (MCM)	+2.24%	+0.08%
Molecular Topological Index (MTI)	-1.26%	+0.74%
Randic (R)	-1.06%	+0.91%
Spectral Radius (ρ)	+0.42%	-0.58%
Wiener (W)	-1.16%	+0.08%
Wiener Hyper (WW)	-2.36%	+0.42%
Wiener Reverse (Λ)	+1.78%	+0.08%
Wiener Terminal (TW)	-2.02%	+1.06%
Zagreb M_1	+1.03%	-0.56%
Zagreb M_2	-1.29%	+0.08%

Table 5.7: Relative change in area under the receiver operating characteristic curve (ROC-AUC \uparrow) on `ogbg-molhiv` dataset, GCN+GF variants, over their respective (global-feature free) baselines (higher is better). The baseline GCN achieves 0.7549 ± 0.016 .

Global feature f	GCN + GF	GCN + GF _{NB}
Algebraic Connectivity (a)	-0.42%	-2.20%
Balaban (J)	+1.17%	-0.95%
Circuit Rank (μ)	+0.78%	-0.32%
Estrada (EE)	+0.62%	-0.99%
Harary index (H)	+1.65%	-0.41%
HOMO-LUMO (R_{HL})	-0.35%	+0.08%
Independence Number (α)	+1.16%	+1.58%
Kirchhoff index (Kf)	+2.44%	-1.96%
Max. card. Matching (MCM)	+1.13%	-0.73%
Molecular Topological Index (MTI)	+3.36%	-1.69%
Randic (R)	+1.41%	-0.70%
Spectral Radius (ρ)	+1.04%	-2.08%
Wiener (W)	+2.48%	-2.15%
Wiener Terminal (TW)	-0.15%	-0.21%
Zagreb M_1	+1.20%	-1.37%
Zagreb M_2	+3.22%	-0.56%

Table 5.8: Relative change in area under the receiver operating characteristic curve (ROC-AUC \uparrow) on `ogbg-molhiv` dataset, GIN+GF variant, over its respective (global-feature free) baseline (higher is better). The baseline GIN achieves 0.7674 ± 0.015 .

Global feature f	GIN + GF
Algebraic connectivity (a)	+0.25%
Circuit rank (μ)	+0.70%
Independence Number (α)	-0.18%
Max. card. Matching (MCM)	+0.03%
Spectral Radius (ρ)	+0.29%
Wiener (W)	+0.04%
Zagreb M_1	+0.09%
Zagreb M_2	+0.61%
All features	+0.29%
$f(G) = 1$	-0.95%

Conclusion

In this thesis, we have investigated the effect of using global features with GNNs. We have shown that global features used with message passing neural networks can increase their expressive power, and can be integrated with any GNN. This method has many advantages. First, MPNNs and other standard message passing models cannot compute certain graph invariants, such as the graph diameter. By easily integrating these global features with MPNNs, we are able to incorporate information that is otherwise inaccessible to the model, thereby enhancing its ability to capture more of the graph data. Secondly, this method can be easily combined with existing GNNs implementations, as it only requires changes to the input graph data.

We have selected a list of 23 global features and proposed a method of classifying and integrating them with the learnt embeddings of a GNN. We have proven that most of them can be used to provably increase the expressivity of the WL algorithm and MPNNs. Only four GFs do not as they are based on graph properties expressible by WL, i.e., the degree sequence and the spectral radius of the adjacency matrix. Furthermore, we demonstrated that they can increase the empirical performance of GNNs on graph-level tasks. For this purpose, we have experimented with three different GNNs: CWN, GCN and GIN on two different datasets: ZINC and ogbg-molhiv.

However, our approach has its challenges and limitations as well. Firstly, we have demonstrated that we can provably increase the expressiveness of MPNNs, such that a MPNN together with a global feature can be strictly more expressive than the MPNN alone. Unfortunately, the expressive power of GNNs is not always relevant to real-world applications. For example, Zopf [2022] has shown that for many graph datasets, the WL test is enough to distinguish almost every graph in the dataset. Moreover, they showed that WL methods can still achieve high predictive performance on most graph datasets. Our experimental results confirm the same hypothesis: we observed that non-expressivity-increasing global features, e.g., the Zagreb indices, perform competitively to the expressivity-increasing features. Secondly, there are a few disadvantages to

adding the global features with the GNN learnt embeddings. Some of the global feature values may overlap or correlate with the information already captured by the GNN, leading to redundancy. There is a risk that the model may start overfitting the global features, especially in the case of the node-broadcasted variant, which has the numerical value associated with the global feature replicated across all of the nodes in the graph. Thirdly, we observed that the effectiveness of the GF variants depends on the choice of global features as well. If the global features are not well-selected or do not contribute meaningfully to the task at hand, they can even degrade the model's performance, as we have seen with GF_{NB} variant on GCN.

Finally, we will now give an overview on possible future work. Firstly, due to the integration method not being specific to a type of GNN, we believe that integrating global features with more recent state-of-the-art graph neural networks, e.g., *CSA* [Menegaux et al., 2023] on the ZINC dataset, is an interesting direction for future work.

Secondly, our work in regards to the global features has been mostly explorative: we have picked a list of features based on different graph properties and evaluated their expressive power and predictive performance together with GNNs. Going further, it would be interesting in finding a heuristic for determining what global features would fit best given a graph-level task. Moreover, given that there are more than 100 global features available [Huuskonen et al., 1998], a dimensionality-reduction technique applied across the whole set of features for a graph dataset could be promising. For example, a PCA over the global features vectors seems reasonable.

List of Figures

1.1	A benzene molecule represented as graph G (ignoring hydrogen atoms), where the nodes represent the atoms of carbon and the edges the bonds between the respective atoms. The GNN can predict certain molecular behaviours in relation to different media: reactivity, solubility and toxicity.	3
2.1	Left: a complete graph with 4 nodes, denoted as K_4 , right: a graph in the shape of a hexagon, denoted as C_6 , resembling a benzene molecule.	5
2.2	A pair of isomorphic graphs (dotted lines show the mapping between the nodes of the two graphs).	8
2.3	The molecules bicyclopentyl (left) and decalin (right) which cannot be distinguished by WL algorithm and thus by any MPNN. Node colours are the result of the WL node colouring algorithm. Figure adapted from Sato [2020].	18
3.1	Conversion of a benzene ring to a molecular fingerprint.	21
4.1	Classification of global features based on graph properties such as distance, degree, eigenvalues, matchings, and mixed.	26
4.2	Randic index for 3 molecules: benzene, ethylene and cyclopropane.	27
4.3	Wiener index for 3 molecules: benzene, ethylene and cyclopropane.	27
4.4	Spectral radius for 3 molecules: benzene, ethylene and cyclopropane.	29
4.5	Hosoya index for 3 molecules: benzene, ethylene and cyclopropane.	30
4.6	Degree Distance index for 3 molecules: benzene, ethylene and cyclopropane.	31
4.7	G_1	34
4.8	G_2	34
4.9	G_3	38
4.10	G_4	38
4.11	G_5	39
4.12	G_6	39
4.13	Our proposed GNN architecture. We concatenate global graph features to the output of the graph pooling layer after multiple GNN layers. Figure adapted from Welke et al. [2023].	41



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Tables

4.1	Comparison of distance-based global features values for graphs G_1 and G_2 .	35
4.2	Comparison of diameter values for graphs G_5 and G_6	35
4.3	Comparison of matching-based global features values for graphs G_1 and G_2 .	36
4.4	Comparison of eigenvalue-based global features values for graphs G_3 and G_4 .	37
4.5	Comparison of mixed global features values for graphs G_1 and G_2	38
4.6	Comparison of mixed global features values for graphs G_3 and G_4	38
4.7	Summary of the global features expressiveness and runtime complexity.	40
5.1	Hyperparameter grid used for the ZINC dataset.	48
5.2	Hyperparameter grid used for the ogbg-molhiv dataset.	48
5.3	Relative change in mean absolute error (MAE ↓) on ZINC dataset, CWN+GF variants, over their respective (global-feature free) baselines (lower is better). The baseline CWN achieves 0.0785 ± 0.003	49
5.4	Relative change in mean absolute error (MAE ↓) on ZINC dataset, GCN+GF variants, over their respective (global-feature free) baselines (lower is better). The baseline GCN achieves 0.1951 ± 0.009	50
5.5	Relative change in mean absolute error (MAE ↓) on ZINC dataset, GIN+GF variant, over its respective (global-feature free) baseline (lower is better). The baseline GIN achieves 0.1854 ± 0.004	51
5.6	Relative change in area under the receiver operating characteristic curve (ROC-AUC ↑) on ogbg-molhiv dataset, CWN+GF variants, over their respective (global-feature free) baselines (higher is better). The baseline CWN achieves 0.7810 ± 0.014	51
5.7	Relative change in area under the receiver operating characteristic curve (ROC-AUC ↑) on ogbg-molhiv dataset, GCN+GF variants, over their respective (global-feature free) baselines (higher is better). The baseline GCN achieves 0.7549 ± 0.016	52
5.8	Relative change in area under the receiver operating characteristic curve (ROC-AUC ↑) on ogbg-molhiv dataset, GIN+GF variant, over its respective (global-feature free) baseline (higher is better). The baseline GIN achieves 0.7674 ± 0.015	52



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Bibliography

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=ryGs6iA5Km>.

Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: higher-order graph neural networks. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19. AAAI Press, 2019. ISBN 978-1-57735-809-1. doi: 10.1609/aaai.v33i01.33014602. URL <https://doi.org/10.1609/aaai.v33i01.33014602>.

Teague Sterling and John J. Irwin. ZINC 15 – Ligand Discovery for Everyone. *Journal of Chemical Information and Modeling*, 55(11):2324–2337, November 2015. ISSN 1549-9596. doi: 10.1021/acs.jcim.5b00559.

Vikas K. Garg, Stefanie Jegelka, and Tommi Jaakkola. Generalization and representational limits of graph neural networks. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Jarmo Huuskonen, Marja Salo, and Jyrki Taskinen. Aqueous Solubility Prediction of Drugs Based on Molecular Topology and Neural Network Modeling. *Journal of Chemical Information and Computer Sciences*, 38(3):450–456, May 1998. ISSN 0095-2338, 1520-5142. doi: 10.1021/ci970100x.

Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yuguang Wang, Pietro Liò, Guido F Montufar, and Michael Bronstein. Weisfeiler and leman go cellular: Cw networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 2625–2640. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/157792e4abb490f99dbd738483e0d2d4-Paper.pdf.

- Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL <https://openreview.net/forum?id=SJU4ayYgl>.
- Andrei Dragos Brasoveanu, Fabian Jogl, Pascal Welke, and Maximilian Thiessen. Extending graph neural networks with global features. In *The Second Learning on Graphs Conference*, 2023.
- Reinhard Diestel. *Graph Theory*, volume 173. 01 2005. doi: 10.1007/b100033.
- R. L. Graham and H. O. Pollak. On the addressing problem for loop switching. *The Bell System Technical Journal*, 50(8):2495–2519, 1971. doi: 10.1002/j.1538-7305.1971.tb02618.x.
- F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.
- László Babai. Graph isomorphism in quasipolynomial time, 2016.
- B Weisfeiler and A Leman. The reduction of a graph to canonical form and the algebra which appears therein. 2(9):12–16, 1968.
- G. Chartrand and P. Zhang. *Chromatic Graph Theory*. Discrete Mathematics and Its Applications. CRC Press, 2008. ISBN 9781584888017. URL https://books.google.at/books?id=_l4CJq46MXwC.
- Stefanie Jegelka. Theory of graph neural networks: Representation and learning, 2022.
- John J Irwin and Brian K Shoichet. ZINC—a free database of commercially available compounds for virtual screening. *Journal of Chemical Information and Modeling*, 45(1):177–182, 2005.
- Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- C.M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 2006. ISBN 9780387310732.
- John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011. URL <http://jmlr.org/papers/v12/duchilla.html>.
- Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.
- Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. ISSN 0893-6080. doi: [https://doi.org/10.1016/0893-6080\(89\)90020-8](https://doi.org/10.1016/0893-6080(89)90020-8). URL <https://www.sciencedirect.com/science/article/pii/0893608089900208>.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry, 2017.

Ryoma Sato. A survey on the expressive power of graph neural networks, 2020.

Lothar Von Collatz and Ulrich Sinogowitz. Spektren endlicher grafen: Wilhelm Blaschke zum 70. Geburtstag gewidmet. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 21(1):63–77, December 1957. ISSN 0025-5858, 1865-8784. doi: 10.1007/BF02941924.

Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(2):298–305, 1973. ISSN 0011-4642, 1572-9141. doi: 10.21136/CMJ.1973.101168.

Derek Lim, Joshua David Robinson, Lingxiao Zhao, Tess Smidt, Suvrit Sra, Haggai Maron, and Stefanie Jegelka. Sign and basis invariant networks for spectral graph representation learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=Q-UHqMorzil>.

Gaurav Rattan and Tim Seppelt. Weisfeiler-leman and graph spectra. In *SODA*, 2023.

Yumeng Song, Xiaohua Li, Fangfang Li, and Ge Yu. Learning from Feature and Global Topologies: Adaptive Multi-View Parallel Graph Contrastive Learning. *Mathematics*, 12(14):2277, July 2024. ISSN 2227-7390. doi: 10.3390/math12142277.

Pablo Barceló, Floris Geerts, Juan Reutter, and Maksimilian Ryschkov. Graph neural networks with local graph parameters. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 25280–25293. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/d4d8d1ac7e00e9105775a6b660dd3cbb-Paper.pdf.

Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M. Bronstein. Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):657–668, January 2023. ISSN 0162-8828, 2160-9292, 1939-3539. doi: 10.1109/TPAMI.2022.3154319.

Pascal Welke, Maximilian Thiessen, Fabian Jogl, and Thomas Gärtner. Expectation-complete graph representations with homomorphisms. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 36910–36925. PMLR, 23–29 Jul 2023. URL <https://proceedings.mlr.press/v202/welke23a.html>.

Jan G. Rittig, Qinghe Gao, Manuel Dahmen, Alexander Mitsos, and Artur M. Schweidtmann. Graph Neural Networks for the Prediction of Molecular Structure–Property Relationships. In *Machine Learning and Hybrid Modelling for Reaction Engineering*:

Theory and Applications. Royal Society of Chemistry, 12 2023. ISBN 978-1-83916-563-4. doi: 10.1039/BK9781837670178-00159. URL <https://doi.org/10.1039/BK9781837670178-00159>.

Naifeng Wen, Guanqun Liu, Jie Zhang, Rubo Zhang, Yating Fu, and Xu Han. A fingerprints based molecular property prediction method using the BERT model. *Journal of Cheminformatics*, 14(1):71, October 2022. ISSN 1758-2946. doi: 10.1186/s13321-022-00650-3.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019. URL <https://api.semanticscholar.org/CorpusID:52967399>.

Fabian Offensperger, Gary Tin, Miquel Duran-Frigola, Elisa Hahn, Sarah Dobner, Christopher W. Am Ende, Joseph W. Strohbach, Andrea Rukavina, Vincenth Brennsteiner, Kevin Ogilvie, Nara Marella, Katharina Kladnik, Rodolfo Ciuffa, Jaimeen D. Majumdar, S. Denise Field, Ariel Bensimon, Luca Ferrari, Evandro Ferrada, Amanda Ng, Zhechun Zhang, Gianluca Degliesposti, Andras Boeszoermyeni, Sascha Martens, Robert Stanton, André C. Müller, J. Thomas Hannich, David Hepworth, Giulio Superti-Furga, Stefan Kubicek, Monica Schenone, and Georg E. Winter. Large-scale chemoproteomics expedites ligand discovery and predicts ligand behavior in cells. *Science*, 384(6694): eadk5864, April 2024. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.adk5864.

Harry Wiener. Structural Determination of Paraffin Boiling Points. *Journal of the American Chemical Society*, 69(1):17–20, January 1947. ISSN 0002-7863, 1520-5126. doi: 10.1021/ja01193a005.

Milan Randic. Characterization of molecular branching. *Journal of the American Chemical Society*, 97(23):6609–6615, November 1975. ISSN 0002-7863, 1520-5126. doi: 10.1021/ja00856a001. URL <https://pubs.acs.org/doi/abs/10.1021/ja00856a001>.

Max Horn, Edward De Brouwer, Michael Moor, Yves Moreau, Bastian Rieck, and Karsten Borgwardt. Topological graph neural networks. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=oxXUMeFwEHd>.

Saima Parveen, Nadeem Ul Hassan Awan, Fozia Bashir Farooq, Rakotondrajao Fanja, and Qurat Ul Ain Anjum. Topological Indices of Novel Drugs Used in Autoimmune Disease Vitiligo Treatment and Its QSPR Modeling. *BioMed Research International*, 2022:1–14, November 2022. ISSN 2314-6141, 2314-6133. doi: 10.1155/2022/6045066.

Pál András Papp, Karolis Martinkus, Lukas Faber, and Roger Wattenhofer. Dropgnn: Random dropouts increase the expressiveness of graph neural networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances*

in *Neural Information Processing Systems*, volume 34, pages 21997–22009. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/b8b2926bd27d4307569ad119b6025f94-Paper.pdf.

Alexander Vasilyev and Dragan Stevanovic. Mathchem: A python package for calculating topological indices. *MATCH Communications in Mathematical and in Computer Chemistry*, 71:657–680, 10 2013.

I. Gutman and N. Trinajstić. Graph theory and molecular orbitals. total π -electron energy of alternant hydrocarbons. *Chemical Physics Letters*, 17(4):535–538, 1972. ISSN 0009-2614. doi: [https://doi.org/10.1016/0009-2614\(72\)85099-1](https://doi.org/10.1016/0009-2614(72)85099-1). URL <https://www.sciencedirect.com/science/article/pii/0009261472850991>.

Milan Randić. Novel molecular descriptor for structure—property studies. *Chemical Physics Letters*, 211(4):478–483, 1993. ISSN 0009-2614. doi: [https://doi.org/10.1016/0009-2614\(93\)87094-J](https://doi.org/10.1016/0009-2614(93)87094-J). URL <https://www.sciencedirect.com/science/article/pii/000926149387094J>.

Alexandru Balaban, Denise Mills, Ovidiu Ivanciuc, and Subhash Basak. Reverse wiener indices. *Croatica Chemica Acta*, 73, 12 2000.

Ivan Gutman, Boris Furtula, and Miroslav Petrović. Terminal wiener index. *Journal of Mathematical Chemistry*, 46:522–531, 04 2009. doi: 10.1007/s10910-008-9476-2.

Ivan Gutman and Boris Furtula. A survey on terminal Wiener index. In Ivan Gutman and Boris Furtula, editors, *Novel Molecular Structure Descriptors — Theory and Applications*, pages 173–190. Univ. Kragujevac, Kragujevac, 2010.

Dejan Plavšić, Sonja Nikolić, Nenad Trinajstić, and Zlatko Mihalić. On the Harary index for the characterization of chemical graphs. *Journal of Mathematical Chemistry*, 12(1): 235–250, December 1993. ISSN 0259-9791, 1572-8897. doi: 10.1007/BF01164638.

Alexandru T. Balaban. Highly discriminating distance-based topological index. *Chemical Physics Letters*, 89(5):399–404, July 1982. ISSN 00092614. doi: 10.1016/0009-2614(82)80009-2.

Izrail Solomonovich Gradshteyn, Iosif Moiseevich Ryzhik, and Alan Jeffrey. *Table of Integrals, Series, and Products*. Academic press, San Diego New York Berkeley [etc.], corr. and enlarged ed edition, 1980. ISBN 978-0-12-294760-5.

Ernesto Estrada. Characterization of 3d molecular structure. *Chemical Physics Letters*, 319(5):713–718, 2000. ISSN 0009-2614. doi: [https://doi.org/10.1016/S0009-2614\(00\)00158-5](https://doi.org/10.1016/S0009-2614(00)00158-5). URL <https://www.sciencedirect.com/science/article/pii/S0009261400001585>.

D. J. Klein and M. Randić. Resistance distance. *Journal of Mathematical Chemistry*, 12 (1):81–95, December 1993. ISSN 0259-9791, 1572-8897. doi: 10.1007/BF01164627.

- Danail Bonchev, Alexandru T. Balaban, Xiaoyu Liu, and Douglas J. Klein. Molecular cyclicity and centrality of polycyclic graphs. I. Cyclicity based on resistance distances or reciprocal distances. *International Journal of Quantum Chemistry*, 50(1):1–20, March 1994. ISSN 0020-7608, 1097-461X. doi: 10.1002/qua.560500102.
- Gašper Jaklič, Patrick Fowler, and Tomaz Pisanski. H1-index of a graph. *Ars Mathematica Contemporanea*, 5, 01 2012. doi: 10.26493/1855-3974.180.65e.
- Haruo Hosoya. Topological Index. A Newly Proposed Quantity Characterizing the Topological Nature of Structural Isomers of Saturated Hydrocarbons. *Bulletin of the Chemical Society of Japan*, 44(9):2332–2339, September 1971. ISSN 0009-2673, 1348-0634. doi: 10.1246/bcsj.44.2332.
- Glenn K. Manacher. Algorithmic graph theory (alan gibbons). *SIAM Review*, 31(1):145–147, 1989. doi: 10.1137/1031028. URL <https://doi.org/10.1137/1031028>.
- Andrey A. Dobrynin and Amide A. Kochetova. Degree Distance of a Graph: A Degree Analog of the Wiener Index. *Journal of Chemical Information and Computer Sciences*, 34(5):1082–1086, September 1994. ISSN 0095-2338. doi: 10.1021/ci00021a008.
- G. Kirchhoff. Ueber die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Vertheilung galvanischer Ströme geführt wird. *Annalen der Physik*, 148(12): 497–508, January 1847. ISSN 0003-3804, 1521-3889. doi: 10.1002/andp.18471481202.
- S. Gupta, M. Singh, and A.K. Madan. Eccentric distance sum: A novel graph invariant for predicting biological and physical properties. *Journal of Mathematical Analysis and Applications*, 275(1):386–401, 2002. ISSN 0022-247X. doi: [https://doi.org/10.1016/S0022-247X\(02\)00373-6](https://doi.org/10.1016/S0022-247X(02)00373-6). URL <https://www.sciencedirect.com/science/article/pii/S0022247X02003736>.
- Vikas Sharma, Reena Goswami, and A. K. Madan. Eccentric Connectivity Index: A Novel Highly Discriminating Topological Descriptor for Structure-Property and Structure-Activity Studies. *Journal of Chemical Information and Computer Sciences*, 37(2): 273–282, March 1997. ISSN 0095-2338. doi: 10.1021/ci960049h.
- Chris Godsil and Gordon Royle. *Algebraic Graph Theory*, volume 207 of *Graduate Texts in Mathematics*. Springer New York, New York, NY, 2001. ISBN 978-0-387-95220-8 978-1-4613-0163-9. doi: 10.1007/978-1-4613-0163-9.
- Harry P. Schultz. Topological organic chemistry. 1. Graph theory and topological indices of alkanes. *Journal of Chemical Information and Computer Sciences*, 29(3):227–228, August 1989. ISSN 0095-2338. doi: 10.1021/ci00063a012.
- V. Arvind, Frank Fuhlbrück, Johannes Köbler, and Oleg Verbitsky. On weisfeiler-leman invariance: Subgraph counts and related graph properties. *Journal of Computer and System Sciences*, 113:42–59, 2020. ISSN 0022-0000. doi: <https://doi.org/10.1016/j.jcss.2020.04.003>. URL <https://www.sciencedirect.com/science/article/pii/S0022000020300386>.

- Fabian Jogl, Maximilian Thiessen, and Thomas Gärtner. Reducing learning on cell complexes to graphs. In *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*, 2022. URL <https://openreview.net/forum?id=HKUxAE-J6lq>.
- Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 22118–22133. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/fb60d411a5c5b72b2e7d3527cfc84fd0-Paper.pdf.
- Rafael Gómez-Bombarelli, Jennifer N. Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. *ACS Central Science*, 4(2):268–276, February 2018. ISSN 2374-7943. doi: 10.1021/acscentsci.7b00572.
- Vijay Prakash Dwivedi, Chaitanya K. Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *J. Mach. Learn. Res.*, 24(1), mar 2024. ISSN 1532-4435.
- Markus Zopf. 1-wl expressiveness is (almost) all you need. *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2022. URL <https://api.semanticscholar.org/CorpusID:247011932>.
- Romain Menegaux, Emmanuel Jehanno, Margot Selosse, and Julien Mairal. Self-attention in colors: Another take on encoding graph structure in transformers. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL <https://openreview.net/forum?id=3dQCNqqv2d>.