A Theoretical and Empirical Investigation into the Equivalence of Graph Neural Networks and the Weisfeiler-Leman Algorithm

From the faculty of Mathematics, Physics, and Computer Science approved for the purpose of obtaining the academic degree of Bachelor of Sciences.

Eric Tillmann Bill

Supervision:

Prof. Dr. rer. nat. Christopher Morris

Informatik 6
RWTH Aachen University

Contents

1 Introduction

Yet to come!

2 Related Work

Yet to come!

3 Preliminaries

First, we introduce a couple of notions and definitions that will be used throughout this thesis. In particular, the definitions will play a crucial role in the theoretical part that follows. We start with some general notations, introduce a general graph definition, and familiarize the reader with the Weisfeiler-Leman algorithm. We will then introduce each framework independently, first the 1-WL+NN and then GNN. At the end, we will briefly introduce important properties of collections of functions computed by both methods.

3.1 General Notation

We first introduce a couple of notations and definitions that will be used throughout the thesis. With $[n]$, we denote the set $\{1, \ldots, n\} \subset \mathbb{N}$ for any $n \in \mathbb{N}$ and with $\{\!\{\ldots\}\!\}$ we denote a multiset which is formally defined as a 2 tuple (X, m) with X being a set of all unique elements and $m : X \to \mathbb{N}_{\geq 1}$ a mapping that maps every element in X to its number of occurrences in the multiset.

3.2 Graphs

A graph G is a 3-tuple $G := (V, E, l)$ that consists of the set of all nodes V, the set of all edges $E \subseteq V \times V$ and a label function $l : M \to \Sigma$ with M being either $V, V \cup E$ or E and $\Sigma \subset \mathbb{N}$ a finite alphabet. Moreover, let $\mathcal{G}$ be the set of all finite graphs. Note, that our definition of the label function allows for graphs with labels either only on the nodes, only on the edges, or on both nodes and edges. Sometimes the values assigned by l are called features, but this is usually only the case when Σ is multidimensional, which we do not cover in this thesis. In addition, although we have defined it this way, the labeling function is optional, and in cases where no labeling function is given, we add the trivial labeling function $f_1 : V(G) \to \{1\}$. Further, G can be either directed or undirected, depending on the definition of E, where $E \subseteq \{(v, u) \mid v, u \in V\}$ defines a directed and $E \subseteq \{(v, u), (u, v) \mid v, u \in V, v \neq u\}$ such that for every $(v, u) \in E$ also $(u, v) \in E$ defines an undirected graph. Additionally, we will use the notation $V(G)$ and $E(G)$ to denote the set of nodes of G and the set of edges of G respectively, as well as l_G to denote the label function of G. With $\mathcal{N}(v)$ for $v \in V(G)$ we denote the set of neighbors of v with $\mathcal{N}(v) := \{u \mid (u, v) \in E(G)\}$.

A coloring of a Graph G is a function $C : V(G) \to \mathbb{N}$ that assigns each node in the graph a color (here a positive integer). Further, a coloring C induces a partition P on the set of nodes, for which we define C^{-1} being the function that maps each color $c \in \mathbb{N}$ to its class of nodes with $C^{-1}(c) = \{v \in V(G) \mid C(v) = c\}$. In addition, we define $h_{G,C}$ as the histogram of graph G with coloring C, that maps every color in the image of C under $V(G)$ to the number of occurrences. In detail, $\forall c \in \mathbb{N} : h_{G,C}(c) := |\{v \in V(G) \mid C(v) = c\}| = |C^{-1}(c)|$

Permutation-invariance and -equivariance

We use S_n to denote the symmetric group over the elements $[n]$ for any $n > 0$. S_n consists of all permutations over these elements. Let G be a graph with $V(G) = [n]$, applying a permutation $\pi \in S_n$ on G, is defined as $G_\pi := \pi \cdot G$ where $V(G_\pi) = \{\pi(1), \ldots, \pi(n)\}$ and $E(G_\pi) = \{(\pi(v), \pi(u)) \mid (v, u) \in E(G)\}$. We will now introduce two key concepts for classifying functions on graphs.

Definition 1 (Permutation Invariant)**.** Let $f : \mathcal{G} \to \mathcal{X}$ be an arbitrary function and let $V(G) = [n]$ for some $n \in \mathbb{N}$. The function f is *permutation-invariant* if and only if for all $G \in \mathcal{G}$ where $n_G := \mid V(G) \mid$ and for every $\pi \in S_{n_G}$: $f(G) = f(\pi \cdot G)$.

Definition 2 (Permuation Equivariant)**.** Let $f : \mathcal{G} \to \mathcal{X}$ be an arbitrary function and let $V(G) = [n]$ for some $n \in \mathbb{N}$. The function f is *permuation-equivariant* if and only if for all $G \in \mathcal{G}$ where $n_G := \mid V(G) \mid$ and for every $\pi \in S_{n_G}$: $f(G) = \pi^{-1} \cdot f(\pi \cdot G)$.

3.3 Weisfeiler and Leman Algorithm

The Weisfeiler-Leman algorithm consists of two main parts, first the coloring algorithm and second the graph isomorphism test. We will introduce them in this section.

The Weisfeiler-Leman graph coloring algorithm

The 1-WL algorithm computes a node coloring of its input graph in each iteration. A color for a node is computed using only the coloring of its neighbors and the node itself. The algorithm will continue as long as it has not converged, and returns the final coloring of the graph.

Definition 3 (1-WL Algorithm)**.** Let $G = (V, E, l)$ be a graph, then in each iteration i, the 1-WL computes a node coloring $C_i : V(G) \to \mathbb{N}$. In iteration $i = 0$, the initial coloring is $C_0 = l$ or if l is non-existing $\forall v \in V(G) : C_0(v) = c$ for an arbitrary constant $c \in \mathbb{N}$. For $i > 0$, the algorithm assigns a color to $v \in V(G)$ as follows:

$$C_i(v) = \mathsf{RELABEL}(C_{i-1}(v), \{\!\!\{ C_{i-1}(u) \mid u \in \mathcal{N}(v) \}\!\!\}),$$

where $\mathsf{RELABEL}$ injectively maps the above pair to a unique, previously not used, natural number. Although this is not a formal restriction by the inventors, we further require the function to always map to the next minimal natural number. Thereby we can contain the size of the codomain of each coloring for all iterations. The algorithm terminates when the number of colors between two iterations does not change, meaning the algorithm terminates after iteration i if the following condition is satisfied:

$$\forall v, w \in V(G) : C_i(v) = C_i(w) \iff C_{i+1}(v) = C_{i+1}(w).$$

Upon terminating we define $C_\infty := C_i$ as the stable coloring, such that $\text{1-WL}(G) := C_\infty$.

The colorings computed in each iteration always converge to the final one, such that the algorithm always terminates. In more detail, Grohe [2017] showed that it always holds after at most $|V(G)|$ iterations. For an illustration of this coloring algorithm, see Figure 2. Moreover, based on the work of Paige and Tarjan [1987] about efficient refinement strategies, Cardon and Crochemore [1982] proved that the stable coloring C_∞ can be computed in time $\mathcal{O}(|V(G)| + |E(G)| \cdot \log |V(G)|)$.

The Weisfeiler-Leman Graph Isomorphism Test

Definition 4 (1-WL Isomorphism Test)**.** To determine if two graphs $G, H \in \mathcal{G}$ are non-isomorphic ($G \not\cong H$), one applies the 1-WL coloring algorithm on both graphs "in parallel" and checks after each iteration if the occurrences of each color are equal, else the algorithm would

terminate and conclude non-isomorphic. Formally, the algorithm concludes non-isomorphic in iteration i if there exists a color c such that:

$$|\{v \in V(G) \mid c = C_i(v)\}| \neq |\{v \in V(H) \mid c = C_i(v)\}|.$$

Note that this test is only sound and not complete for the *graph isomorphism problem*. Counterexamples where the algorithm fails to distinguish non-isomorphic graphs can be easily constructed, see Figure 1 which was discovered and proven by Cai et al. [1992].
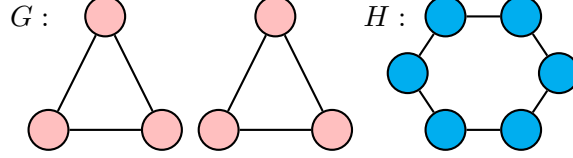

Figure 1: An example of two graphs G and H that are non-isomorphic but cannot be distinguished by the 1-WL isomorphism test.

3.4 1-WL+NN

As seen in the previous section, the 1-WL algorithm is quite powerful in identifying substructures. With the 1-WL+NN framework, we define functions that utilize this structural information to derive further application-specific insights. We do this by combining well-known machine learning techniques and the algorithm.

Definition 5 (1-WL+NN)**.** We say the function $\mathcal{B} : \mathcal{G} \to \mathbb{R}^m$ is computable by 1-WL+NN, if it can be compromised as $\mathcal{B}(\cdot) = \text{MLP} \circ f_{\text{enc}} \circ \text{1-WL}(\cdot)$, where f_{enc} is a permutation invariant encoding function that maps graph-colorings to fixed-sized vectors, and MLP is a multilayer perceptron.

As a concrete example of a collection of functions computable by 1-WL+NN we will introduce the collection $\mathfrak{B}_k$ that is parametrized by $k \in \mathbb{N}_{\geq 1}$. All functions $\mathcal{B} \in \mathfrak{B}_k$ use the *counting-encoding* function f_{count} as their encoding function, and are constrained in their domain to only work over a subset $\mathcal{X}$ of $\mathcal{G}$. We will define this particular encoding function in the following:

Definition 6 (Counting Encoding Functions)**.** For $k \in \mathbb{N}_{\geq 1}$, let

$$\mathcal{X} = \{G \in \mathcal{G} \mid \forall x \in V(G) \cup E(G) : l_G(x) \leq k\} \subset \mathcal{G}$$

be the set of all graphs, where the label alphabet Σ of the respective label function l is bounded with $\Sigma \subseteq [k]$. We define the *counting-encoding* function $f_{\text{count}} : \text{1-WL}(\mathcal{X}) \to \mathbb{N}^K$ as the function that maps a graph coloring C_∞ of a graph $G \in \mathcal{X}$ to a vector $v \in \mathbb{N}^K$ such that the c.th component of v is equal to the number of occurrences of the color c in the coloring C_∞. More formally, for $G \in \mathcal{X}$ let C_∞ be the final coloring upon the termination of the 1-WL algorithm on G and h_{G,C_∞} the respective color histogram. Then f_{count} maps C_∞ to a vector $v \in \mathbb{N}^K$, such that for all $c \in [K] : v_c = h_{G,C_\infty}(c)$, where v_c denotes the c.th component of the vector v. Important to note, due to the bounded label alphabet Σ of all graphs $G \in \mathcal{X}$ by the parameter k, there exists a minimal K for the codomain $\mathbb{N}^K$ of f_{count}, such that f_{count} is well-defined on all graphs $G \in \mathcal{X}$.

To illustrate how this encoding function works and why we coined it *counting-encoding*, we will quickly introduce an example graph G. In Figure 2, we give a visual representation of G and its stable coloring after applying the 1-WL algorithm to it. The *counting-encoding* function f_{count} counts through all colors $i \in [K]$ and sets each i.th component of the output vector to the number of occurrences in the final coloring. Therefore, the respective color histogram $h_{G,C_\infty} = \{\!\{2, 2, 3, 4\}\!\}$ of G is being mapped to $v \in \mathbb{N}^K$ with $v = (0, 2, 1, 1, 0, \ldots, 0)^T$, since color 2 appears two times, while color 3 and 4 occur only once. All other components of v are set to 0.
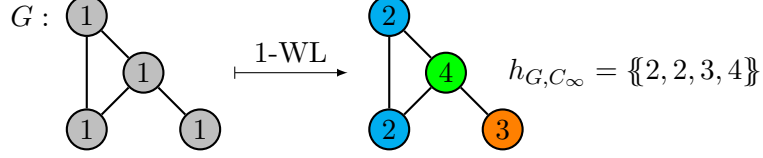

Figure 2: An example of the final coloring computed by applying the 1-WL algorithm on the graph G. The graph G consists of 4 nodes with all their labels being initially set to 1. Note that each label corresponds to a color, which we have also plotted for illustration purposes.

3.5 Graph Neural Networks (Message Passing)

A Graph Neural Network (GNN) is a composition of multiple layers, where each layer computes a new feature for each node and edge. The GNN layer thus technically obtains a new graph that is structurally identical to the previous one, but contains new feature information. After an input graph has been passed through all layers, there can be an additional final function, aggregating the computed information into a fixed size output. With this, it is possible to apply a GNN to every graph, regardless of its size, as the "computation" will only take place on the nodes and edges of the graph.

Note that in the following we will restrict the definition to only consider node features, however, one can easily extend it to also include edge features.

Definition 7 (Graph Neural Network)**.** Let $G = (V, E, l)$ be an arbitrary graph. A Graph Neural Network (GNN) is a composition of multiple layers where each layer t is represented by a function $f^{(t)}$ that works over the set of nodes $V(G)$. To begin with, we need a function $f^{(0)}$: $V(G) \to \mathbb{R}^{1 \times d}$ that is consistent with l, that translates all labels into a vector representation. Further, for every $t > 0$, $f^{(t)}$ is of the format:

$$f^{(t)}(v) = f_{\text{merge}}^{W_{1,t}}(f^{(t-1)}(v),\ f_{\text{agg}}^{W_{2,t}}(\{\!\{f^{(t-1)}(w) \mid w \in \mathcal{N}(v)\}\!\})),$$

where $f_{\text{merge}}^{W_{1,t}}$ and $f_{\text{agg}}^{W_{2,t}}$ are arbitrary differentiable functions with $W_{1,t}$ and $W_{2,t}$ their respective parameters. Additionally, $f_{\text{agg}}^{W_{2,t}}$ has to be permuation-invariant.

Depending on the objective, whether the GNN is tasked with a graph or a node task, the last layer differs. In the case of graph tasks, we add a permutation-invariant aggregation function to the end, here called READOUT, that aggregates over every node and computes a fixed-size output vector for the entire graph, e.g. a label for graph classification. In order to ensure that we can train the GNN in an end-to-end fashion, we require READOUT to be also differentiable. Let $\mathcal{A}$ be an instance of the described GNN framework. Further, let $K \in \mathbb{N}$ be the number

of layers of the GNN, $\mathcal{G}$ the set of all graphs, $\mathcal{Y}$ the task-specific output set (e.g. labels of a classification task), then the overall function computed by $\mathcal{A}$ is:

$$\mathcal{A} : \mathcal{G} \to \mathcal{Y} : x \mapsto \mathsf{READOUT} \circ f^{(K)} \circ \ldots \circ f^{(0)}(x),$$

if $\mathcal{A}$ is configured for a graph task, otherwise:

$$\mathcal{A} : \mathcal{G} \to \mathcal{Y} : x \mapsto f^{(K)} \circ \ldots \circ f^{(0)}(x).$$

Note that, as we require all aggregation functions to be permutation-invariant, the total composition $\mathcal{A}$ is permutation-invariant, and with similar reasoning, it is also differentiable. This enables us to train $\mathcal{A}$ like any other machine learning method in an end-to-end fashion, regardless of the underlying encoding used for graphs. This definition and use of notation are inspired by Morris et al. [2019] and Xu et al. [2019].

To demonstrate what kind of functions are typically used, we provide functions used by Hamilton et al. [2017] for a node classification:

$$f_{\mathrm{merge}}^{W_{1,t}}(v) = \mathrm{ReLU}(W_{\mathrm{merge}} \cdot \mathsf{concat}(f^{(t-1)}(v), \ f_{\mathrm{agg}}^{W_{2,t}}(v)))$$

$$f_{\mathrm{agg}}^{W_{2,t}}(v) = \mathsf{MAX}(\{\mathrm{ReLU}(W_{\mathrm{pool}} \cdot f^{(t-1)}(u) + b) \mid u \in \mathcal{N}(v)\})$$

where ReLU is a non-linear element wise activation function, MAX the element-wise max operator; $W_{\mathrm{merge}}, W_{\mathrm{pool}}$ are trainable matrices, b a trainable vector and concat the concatenation function.

3.6 Important for later

In this section, we introduce a formal definition of multilayer perceptron as it is required in a later proof, as well as the $\simeq_{1\mathrm{WL}}$ relation. Additionally, two very important properties for collections of functions.

Definition 8 (Multilayer Perceptron)**.** Multilayer perceptrons are a class of functions from $\mathbb{R}^n$ to $\mathbb{R}^m$, with $n, m \in \mathbb{N}$. In this thesis, we define a multilayer perceptron as a finite sequence, such that a multilayer perceptron MLP is defined as $\mathrm{MLP} := (\mathrm{MLP})_{i \in [k]}$ where k is the number of layers. For every $i \in [k]$, the i.th layer of the MLP is the i.th item in the finite sequence $(\mathrm{MLP})_i$. Further, all layers are recursively defined as:

$$(\mathrm{MLP})_0(v) := v$$
$$(\mathrm{MLP})_{i+1}(v) := \sigma(W_i \cdot (\mathrm{MLP})_i(v) + b_i), \quad \forall i \in [k-1]$$

where σ is an element wise activation function, W_i is the weight matrix and b_i the bias vector of layer i. Note, that for each W_i, the succeeding W_{i+1} must have the same number of columns as W_i has rows, in order to be well-defined. Similarly, for every layer i, W_i and b_i have to have the same number of rows. Following this definition, when applying a MLP on input $v \in \mathbb{R}^n$ it is $\mathrm{MLP}(v) := (\mathrm{MLP})_k(v)$.

Definition 9 (1-WL Relation)**.** For any graphs $G, H \in \mathcal{X}$ we will denote $G \simeq_{1\mathrm{WL}} H$ if the 1-WL isomorphism test can not distinguish both graphs. Note that due to the soundness of this algorithm, if $G \not\simeq_{1\mathrm{WL}} H$, we always can conclude that $G \not\simeq H$.

The $\simeq_{1\mathrm{WL}}$ relation can further be classified as an equivalence relation, as it is reflexive, symmetric and transitive. With this, we introduce a notation of its equivalence classes. Let $G \in \mathcal{X}$, then we denote with $\mathcal{X}/\!\simeq_{1\mathrm{WL}}(G) := \{G' \in \mathcal{X} \mid G \simeq_{1\mathrm{WL}} G'\}$ its equivalence class.

Definition 10 (1-WL-Discriminating)**.** Let $\mathcal{C}$ be a collection of permutation invariant functions from $\mathcal{X}$ to $\mathbb{R}$. We say $\mathcal{C}$ is **1-WL-Discriminating** if for all graphs $G_1, G_2 \in \mathcal{X}$ for which the 1-WL isomorphism test concludes non-isomorphic ($G_1 \not\simeq_{1\mathrm{WL}} G_2$), there exists a function $h \in \mathcal{C}$ such that $h(G_1) \neq h(G_2)$.

Definition 11 (GNN-Approximating)**.** Let $\mathcal{C}$ be a collection of permutation invariant functions from $\mathcal{X}$ to $\mathbb{R}$. We say $\mathcal{C}$ is **GNN-Approximating** if for all permutation-invariant functions $\mathcal{A}$ computed by a GNN, and for all $\epsilon \in \mathbb{R}$ with $\epsilon > 0$, there exists $h_{\mathcal{A},\epsilon} \in \mathcal{C}$ such that $\|\mathcal{A} - h_{\mathcal{A},\epsilon}\|_\infty := \sup_{G \in \mathcal{X}} |\mathcal{A}(G) - h_{\mathcal{A},\epsilon}(G)| < \epsilon$

4 Theoretical Connection

This section is the main part of our theoretical investigation of the two frameworks. We will present 4 intriguing theorems, which will be proven separately afterwards. These results will form the basis for the empirical part that follows. The first two theorems will establish an equivalence between the two frameworks when the input set of graphs is finite. The last two theorems will go one step further and establish a connection for continuous functions computed by 1-WL+NN and GNNs and prove a somewhat weaker connection between them.

Throughout the first two theorems we will concentrate on a finite collection of graphs which we will denote with $\mathcal{X} \subset \mathcal{G}$.

Theorem 12 (Finite Case: "GNN $\subseteq$ 1-WL+NN")**.** Let $\mathcal{C}$ be a collection of functions from $\mathcal{X}$ to $\mathbb{R}$ computable by GNNs, then $\mathcal{C}$ is also computable by 1-WL+NN.

Theorem 13 (Finite Case: "1-WL+NN $\subseteq$ GNN")**.** Let $\mathcal{C}$ be a collection of functions from $\mathcal{X}$ to $\mathbb{R}$ computable by 1-WL+NN, then $\mathcal{C}$ is also computable by GNNs.

With these theorems, we showed the equivalence between both frameworks such that every function computed by 1-WL+NN is also computable by a GNN, and vice versa.

Notice that, we didn't leverage any constraints on the encoding of graphs throughout the first two theorems and their corresponding proves, but rather kept it general. In order to investigate the relation between both frameworks for continuous features spaces, we will first introduce an encoding of graphs that will be used throughout the proof of both the following theorems.

Definition 14. Let X be a compact subset of $\mathbb{R}$ including 0. We decode graphs with n nodes as a matrix $G \in X^{n \times n}$, where $G_{i,i}$ decodes the label of node i for $i \in [n]$, and $G_{i,j}$ with $i \neq j \in [n]$ decodes an edge from node i to j and a corresponding edge features. We say that there is an edge between node i and j if and only if $G_{i,j} \neq 0$. Further, if G encodes an undirected graph, G is symmetric. For simplicity, we denote $\mathcal{X} := X^{n \times n}$ throughout the next two theorems and their respective proofs.

OPEN FOR NOW: Can 1-WL+NN even be applied on continuous graphs, since the 1-WL algorithm seems to only work with discrete color values. Two things why I still believe it is possible, first Morris et al. [2016] developed a Weisfeiler Leman subtree kernel working on continuous graphs. Secondly, in my case, graph attributes are always drawn from a compact subset of $\mathbb{R}$, such that one can define the RELABEL of the 1-WL algorithm to work over $\mathbb{R}$. !

Theorem 15 (Continuous Case: "GNN $\subseteq$ 1-WL+NN"). Let $\mathcal{C}$ be a collection of continuous functions from $\mathcal{X}$ to $\mathbb{R}$ computable by 1-WL+NN. If $\mathcal{C}$ is 1-WL-Discriminating, then there exists a collection of functions $\mathcal{C}'$ computable by 1-WL+NN that is GNN-Approximating.

Theorem 16 (Continuous Case: "1-WL+NN $\subseteq$ GNN"). Let $\mathcal{C}$ be a collection of continuous functions from $\mathcal{X}$ to $\mathbb{R}$ that is GNN-Approximating, then $\mathcal{C}$ is also 1-WL-Discriminating.

Immediately from the last theorem follows the corollary:

Corollary 17. There exists a collection $\mathcal{C}$ of function from $\mathcal{X}$ to $\mathbb{R}$ computable by GNNs that is 1-WL-Discriminating.

Proof. The collection of all functions computable by GNNs is trivially GNN-Approximating, such that we can use Theorem 16 which concludes the proof. $\square$

By putting both theorems into perspective, we can now argue that even for continuous functions 1-WL+NN and GNNs can compute almost the same functions. Each framework can approximate the other framework arbitrarily well. From this we can conclude that the ability 1-WL-Discriminating is very powerful, so we can assume that 1-WL+NN is sufficiently expressive for the upcoming empirical part.

4.1 Proof of Theorem 12

We will prove Theorem 12 by introducing a couple of small lemmas, which combined prove the theorem. In detail, in Lemma 18 we show the existence of a collection computed by 1-WL+NN that is 1-WL-Discriminating. In Lemmas 19 to 21 we derive properties of 1-WL+NN functions we will use throughout Lemmas 22 to 24 with which we prove the theorem. We took great inspiration for Lemmas 22 to 24 from the proof presented in section 3.1 in the work of Chen et al. [2019].

Lemma 18. There exists a collection $\mathcal{C}$ of functions from $\mathcal{X}$ to $\mathbb{R}$ computable by 1-WL+NN that is 1-WL-Discriminating.

Proof. We consider the collection $\mathfrak{B}_k$ (Definition 6) of functions from $\mathcal{X}$ to $\mathbb{R}$ computed by 1-WL+NN, where we choose k as follows:

$$k := \max(\{l_G(v) \mid G \in \mathcal{X}, v \in V(G)\}),$$

the largest label of any node of any graph in $\mathcal{X}$. Note that we can compute k, since $\mathcal{X}$ is finite.

Let $G_1, G_2 \in \mathcal{X}$ such that the 1-WL isomorphism test concludes non-isomorphic ($G_1 \not\simeq_{1\text{WL}} G_2$). Let C_1, C_2 be the final colorings computed by the 1-WL algorithm when applied on G_1, G_2 respectively. Due to $G_1 \not\simeq_{1\text{WL}} G_2$, there exists a color $c \in \mathbb{N}$ such that $h_{G_1,C_1}(c) \neq h_{G_2,C_2}(c)$. If we now consider as multilayer perceptron $\text{MLP}_c : \mathbb{N}^K \to \mathbb{R}, v \mapsto W \cdot v$ with $W \in \mathbb{N}^{1 \times K}$ such that $W_{1,c} := 1$ and $W_{1,i} := 0$ for all $i \in [K] \setminus \{c\}$ (Note, K is a constant introduced by f_{count}). We can construct $\mathcal{B}(\cdot) := \text{MLP}_c \circ f_{\text{count}} \circ \text{1-WL}(\cdot)$, such that $\mathcal{B} \in \mathfrak{B}_k$. Then $\mathcal{B}(G_i) := h_{G,C_i}(c)$ for $i \in \{1, 2\}$, such that we can conclude $\mathcal{B}(G_1) \neq \mathcal{B}(G_2)$, and since G_1, G_2 were arbitrary chosen, we can conclude the proof. $\square$

Lemma 19 (1-WL+NN Equivalence). Let $\mathcal{B}$ be a function over $\mathcal{X}$ computable by 1-WL+NN, then for every pair of graphs $G_1, G_2 \in \mathcal{X}$: if $G_1 \simeq_{1\text{WL}} G_2$ than $\mathcal{B}(G_1) = \mathcal{B}(G_2)$.

Proof. Let $\mathcal{B}$ be an arbitrary function over $\mathcal{X}$ computable by 1-WL+NN, then $\mathcal{B}$ is composed as follows: $\mathcal{B}(\cdot) = \text{MLP} \circ f_{\text{enc}} \circ \text{1-WL}(\cdot)$. Further, let $G_1, G_2 \in \mathcal{X}$ be arbitrary graphs with $G_1 \simeq_{\text{1WL}} G_2$, then by definition of the relation $\simeq_{\text{1WL}}$ we know that $\text{1-WL}(G_1) = \text{1-WL}(G_2)$. With this the equivalence follows immediately. $\qquad\square$

Lemma 20 (1-WL+NN Permuation Invariance)**.** Let $\mathcal{B}$ be a function over $\mathcal{X}$ computable by 1-WL+NN, then $\mathcal{B}$ is permutation invariant.

Proof. Let $G_1, G_2 \in \mathcal{X}$ be arbitrary graphs with $G_1 \simeq G_2$ and $\mathcal{B}$ an arbitrary function computable by 1-WL+NN. Since the 1-WL algorithm is sound, we know that $G_1 \simeq G_2$ implies $G_1 \simeq_{\text{1WL}} G_2$. Using Lemma 19, we can therefore conclude that: $\mathcal{B}(G_1) = \mathcal{B}(G_2)$. $\qquad\square$

Lemma 21 (1-WL+NN Composition)**.** Let $\mathcal{C}$ be a collection of functions computable by 1-WL+NN. Further, let $h_1, \ldots h_n \in \mathcal{C}$ and $\text{MLP}^\bullet$ an multilayer perceptron, than the function $\mathcal{B}$ composed of $\mathcal{B}(\cdot) := \text{MLP}^\bullet(h_1(\cdot), \ldots, h_n(\cdot))$ is also computable by 1-WL+NN.

Proof. Assume the above and let $f_1, \ldots, f_n$ be the encoding functions, as well as $\text{MLP}_1, \ldots, \text{MLP}_n$ be the multilayer perceptrons used by $h_1, \ldots h_n$ respectively. The idea of this proof is, we construct an encoding function f^* that maps a coloring C_∞ to a concatenation of the vectors obtained when applying each encoding function f_i individually. Additionally, we construct a multilayer perceptron MLP^* that takes in this concatenation of vectors and simulates all $\text{MLP}_1, \ldots, \text{MLP}_n$ simultaneously on their respective section of the encoding vector of f^*, and applies afterwards the given $\text{MLP}^\bullet$ on the concatenation of the output of all MLP_i's. See Figure 3 for a sketch of the proof idea. A complete proof can be found in the Appendix, as this proof is very technical and not that interesting.
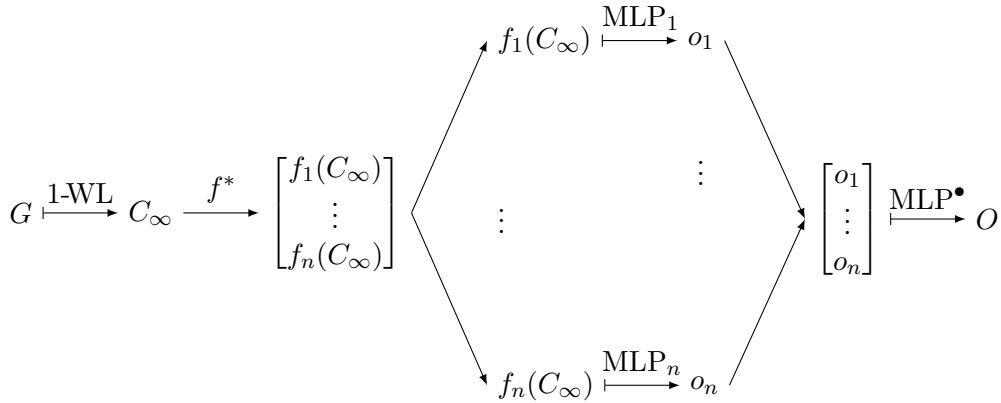

Figure 3: Proof idea for Lemma 21, how the constructed functions f^* and MLP^* will work on input G. Here we denote with C_∞ the final coloring computed by the 1-WL algorithm when applied on G. Further, we let o_i be the output computed by MLP_i on input $f_i(C_\infty)$.

$\qquad\square$

Lemma 22. Let $\mathcal{C}$ be a collection of functions from $\mathcal{X}$ to $\mathbb{R}$ computable by 1-WL+NN that is 1-WL-Discriminating. Then for all $G^* \in \mathcal{X}$, there exists a function h_{G^*} from $\mathcal{X}$ to $\mathbb{R}$ computable by 1-WL+NN, such that for all $G \in \mathcal{X} : h_{G^*}(G) = 0$ if and only if $G \simeq_{\text{1WL}} G^*$.

Proof. Assume the above. For any $G_1, G_2 \in \mathcal{X}$ with $G_1 \not\simeq_{\text{1WL}} G_2$ let $h_{G_1, G_2} \in \mathcal{C}$ be the function distinguishing them, with $h_{G_1, G_2}(G_1) \neq h_{G_1, G_2}(G_2)$. We define the function $\overline{h}_{G_1, G_2}$ working over $\mathcal{X}$ as follows:

$$
\begin{aligned}
\overline{h}_{G_1, G_2}(\cdot) &= |h_{G_1, G_2}(\cdot) - h_{G_1, G_2}(G_1)| \\
&= \max(h_{G_1, G_2}(\cdot) - h_{G_1, G_2}(G_1)) + \max(h_{G_1, G_2}(G_1) - h_{G_1, G_2}(\cdot))
\end{aligned}
\tag{0.1}
$$

Note, that in the formula above "$h_{G_1, G_2}(G_1)$" is a fixed constant and the resulting function $\overline{h}_{G_1, G_2}$ is non-negative. Let $G_1 \in \mathcal{X}$ now be fixed, we will construct the function h_{G_1} with the desired properties as follows:

$$
h_{G_1}(x) = \sum_{G_2 \in \mathcal{X}, \ G_1 \not\simeq_{\text{1WL}} G_2} \overline{h}_{G_1, G_2}(x).
$$

Since $\mathcal{X}$ is finite, the sum is finite and therefore well-defined. Next, we will prove that for a fixed graph $G_1 \in \mathcal{X}$, the function h_{G_1} is correct on input $G \in \mathcal{X}$:

1. If $G_1 \simeq_{\text{1WL}} G$, then for every function $\overline{h}_{G_1, G_2}$ of the sum with $G_1 \not\simeq_{\text{1WL}} G_2$, we know, using Lemma 19, that $\overline{h}_{G_1, G_2}(G)$ is equal to $\overline{h}_{G_1, G_2}(G_1)$ which is by definition 0, such that $h_{G_1}(G) = 0$.

2. If $G_1 \not\simeq_{\text{1WL}} G$, then $\overline{h}_{G_1, G}(G)$ is a summand of the overall sum, and since $\overline{h}_{G_1, G}(G) > 0$, we can conclude $h_{G_1}(G) > 0$ due to the non-negativity of each function $\overline{h}_{G_1, G_2}$.

This function can be encoded via a multilayer perceptron by replacing the max operator in Equation 0.1 by the activation function ReLU. Therefore, we can conclude with Lemma 21 that for every graph G, h_G is also 1-WL+NN computable. $\qquad\square$

Lemma 23. Let $\mathcal{C}$ be a collection of functions from $\mathcal{X}$ to $\mathbb{R}$ computable by 1-WL+NN so that for all $G^* \in \mathcal{X}$, there exists $h_{G^*} \in \mathcal{C}$ satisfying $h_{G^*}(G) = 0$ if and only if $G \simeq_{\text{1WL}} G^*$ for all $G \in \mathcal{X}$. Then for every $G^* \in \mathcal{X}$, there exists a function φ_{G^*} computable by 1-WL+NN such that for all $G \in \mathcal{X}$: $\varphi_{G^*}(G) = \mathbb{1}_{G \simeq_{\text{1WL}} G^*}$.

Proof. Assuming the above. Due to $\mathcal{X}$ being finite, we can define for every graph G^* the constant:

$$
\delta_{G^*} := \frac{1}{2} \min_{G \in \mathcal{X}, \ G \not\simeq_{\text{1WL}} G^*} |h_{G^*}(G)| > 0.
$$

With this constant, we can use a so-called "bump" function working from $\mathbb{R}$ to $\mathbb{R}$ that will be similar to the indicator function. We define this function for parameter $a \in \mathbb{R}$ with $a > 0$ as:

$$
\psi_a(x) := \max(\frac{x}{a} - 1, \ 0) + \max(\frac{x}{a} + 1, \ 0) - 2 \cdot \max(\frac{x}{a}, \ 0).
$$

The interesting property of ψ_a is that it maps every value x to 0, except when x is being drawn from the interval $(-a, a)$. In particular, it maps x to 1 if and only if x is equal to 0. See Figure 7 in the Appendix for a plot of the relevant part of this function with exemplary values for a.

We use these properties to define for every graph $G^* \in \mathcal{X}$ the function $\varphi_{G^*}(\cdot) := \psi_{\delta_{G^*}}(h_{G^*}(\cdot))$. We will quickly demonstrate that this function is equal to the indicator function, for this let G^* be fixed and G, an arbitrary graph from $\mathcal{X}$, the input:

1. If $G \simeq_{\text{1WL}} G^*$, then $h_{G^*}(G) = 0$ resulting in $\varphi_{G^*}(G) = \psi_{\delta_{G^*}}(0) = 1$.

2. If $G \not\simeq_{\text{1WL}} G^*$ then $h_{G^*}(G) \neq 0$, such that $|h_{G^*}(G)| > \delta_{G^*}$ so that $h_{G^*}(G) \notin (-\delta_{G^*}, \delta_{G^*})$ resulting in $\varphi_{G^*}(G) = 0$.

Note that we can encode φ_{G^*} via a multilayer perceptron with a single layer, where δ_{G^*} is a constant and the max operator is replaced by the non-linear activation function ReLU. With Lemma 21 we can therefore conclude that φ_{G^*} is computable by 1-WL+NN for every graph $G^* \in \mathcal{X}$. $\qquad\square$

Lemma 24. Let $\mathcal{C}$ be a collection of functions from $\mathcal{X}$ to $\mathbb{R}$ computable by 1-WL+NN so that for all $G^* \in \mathcal{X}$, there exists $\varphi_{G^*} \in \mathcal{C}$ satisfying $\forall G \in \mathcal{X} : \varphi_{G^*}(G) = \mathbb{1}_{G \simeq_{1\mathrm{WL}} G^*}$, then every function computable by a GNN is also computable by 1-WL+NN.

Proof. Assume the above. For any function $\mathcal{A}$ computed by an GNN that works over $\mathcal{X}$ to $\mathbb{R}$, we show that it can be decomposed as follows for any $G \in \mathcal{X}$ as input:

$$
\begin{aligned}
\mathcal{A}(G) &= \Big(\frac{1}{|\mathcal{X}/\simeq_{1\mathrm{WL}}(G)|} \sum_{G^* \in \mathcal{X}} \mathbb{1}_{G \simeq_{1\mathrm{WL}} G^*} \Big) \cdot \mathcal{A}(G) \\
&= \frac{1}{|\mathcal{X}/\simeq_{1\mathrm{WL}}(G)|} \sum_{G^* \in \mathcal{X}} \mathcal{A}(G^*) \cdot \mathbb{1}_{G \simeq_{1\mathrm{WL}} G^*} \\
&= \sum_{G^* \in \mathcal{X}} \frac{\mathcal{A}(G^*)}{|\mathcal{X}/\simeq_{1\mathrm{WL}}(G^*)|} \cdot \varphi_{G^*}(G) \qquad (0.2)
\end{aligned}
$$

with $\mathcal{X}/\simeq_{1\mathrm{WL}}(G^*)$ we denote the set of all graphs G over $\mathcal{X}$ that are equivalent to G^* according to the $\simeq_{1\mathrm{WL}}$ relation.

Since $\mathcal{A}$ is permutation-invariant, and GNNs are at most as good as the 1-WL algorithm in distinguishing non-isomorphic graphs, we can use the fact that for every graph $G, H \in \mathcal{X}$ with $G \simeq_{1\mathrm{WL}} H$: $\mathcal{A}(G) = \mathcal{A}(H)$. Therefore, we can decompose $\mathcal{A}$ as stated in Equation 0.2 via a multilayer perceptron with $\frac{\mathcal{A}(G^*)}{|\mathcal{X}/\simeq_{1\mathrm{WL}}(G^*)|}$ being constants and $\varphi_{G^*} \in \mathcal{C}$ encoding the indicator function. Combined with the Lemma 21, we can conclude that $\mathcal{A}$ is computable by 1-WL+NN. Important to note, we can only do this since $\mathcal{X}$ is finite, making the overall sum finite and the cardinality of $\mathcal{X}/\simeq_{1\mathrm{WL}}(G^*)$ well-defined for all graphs. $\qquad\square$

4.2 Proof of Theorem 13

In this section we will shortly prove the converse direction. However, the proof is not as interesting as the previous one, as it is mainly just a consequence of Definition 7 of GNNs.

Proof of Theorem 13. Let $\mathcal{C}$ be a collection of functions from $\mathcal{X}$ to $\mathbb{R}$ computable by 1-WL+NN. For any $\mathcal{B} \in \mathcal{C}$ we know, by using Lemma 20 that $\mathcal{B}$ is permutation invariant such that it qualifies as a READOUT function for GNNs. By constructing a GNN with 0 layers and READOUT$(\cdot) := \mathcal{B}(\cdot)$, we obtain a function $\mathcal{A}$ that is computed by this GNN. Moreover, $\mathcal{A}$ is equivalent to $\mathcal{B}$, proving that every function in $\mathcal{C}$ is computable by GNNs. $\qquad\square$

4.3 Proof of Theorem 15

We will prove Theorem 15 by first introducing a definition and then 2 lemmas using that definition which collectively prove the entire theorem. In particular, we define the property that a collection of functions is able to find any $\simeq_{1\mathrm{WL}}$ equivalence class. Then, in Lemma 26, we prove that 1-WL-Discriminating somehow implies being able to locate any $\simeq_{1\mathrm{WL}}$-equivalence class. And finally, in Lemma 27, we prove that being able to locate any $\simeq_{1\mathrm{WL}}$ equivalence class further implies being somehow GNN-approximating. The overall proof is inspired by the work of Chen et al. [2019].

Definition 25 (Locating $\simeq_{\text{1WL}}$ equivalence classes). Let $\mathcal{C}$ be a collection of continuous functions from $\mathcal{X}$ to $\mathbb{R}$. If for all $\epsilon \in \mathbb{R}$ with $\epsilon > 0$ and for every graph $G^* \in \mathcal{X}$ the function h_{G^*} exists in $\mathcal{C}$, with the following properties:

1. for all $G \in \mathcal{X} : h_{G^*}(G) \geq 0$,

2. for all $G \in \mathcal{X}$ with $G \simeq_{\text{1WL}} G^* : h_{G^*}(G) = 0$, and

3. there exists a constant $\delta_{G^*} > 0$, such that for all $G \in \mathcal{X}$ with $h_{G^*}(G) < \delta_{G^*}$ there exists a graph $G' \in \mathcal{X}/\simeq_{\text{1WL}}(G)$ in the equivalence class of G such that $\|G' - G^*\|_2 < \epsilon$

we say $\mathcal{C}$ is able to locate every $\simeq_{\text{1WL}}$ equivalence class.

One can interpret this function h_{G^*} as a kind of loss function that measures the similarity between its input graph to G^*. It yields no loss for the input G, if G is indistinguishable from G^* by the 1-WL algorithm ($G \simeq_{\text{1WL}} G^*$), only a small loss if G is close to a graph in the $\simeq_{\text{1WL}}$ equivalence class of G^* (the loss is upper bounded by δ_{G^*}), and an arbitrary loss otherwise.

Lemma 26. Let $\mathcal{C}$ be a collection of continuous functions from $\mathcal{X}$ to $\mathbb{R}$ computable by 1-WL+NN. If $\mathcal{C}$ is 1-WL-Discriminating, then there exists a collection of functions $\mathcal{C}'$ computable by 1-WL+NN that is able to locate every $\simeq_{\text{1WL}}$ equivalence class on $\mathcal{X}$.

Proof. Let $G^* \in \mathcal{X}$ be fixed and $\epsilon > 0$ be given. Since $\mathcal{C}$ is 1-WL-Discriminating, we know that for every $G \in \mathcal{X}$ with $G \not\simeq_{\text{1WL}} G^*$, there exists a function $h_{G,G^*} \in \mathcal{C}$ such that $h_{G,G^*}(G) \neq h_{G,G^*}(G^*)$. We use this property to construct for each $G \in \mathcal{X}$ with $G \not\simeq_{\text{1WL}} G^*$ a set A_{G,G^*} as follows:

$$A_{G,G^*} := \{G' \in \mathcal{X} \mid h_{G,G^*}(G') \in (h_{G,G^*}(G) \pm \frac{|h_{G,G^*}(G) - h_{G,G^*}(G^*)|}{2})\},$$

where $(a \pm b)$ is the open set $(a - b, a + b)$ over $\mathbb{R}$. One can use Figure 4 below for a better understanding, when a graph G' is contained in A_{G,G^*} and when not. With the illustration one can easily see, that $G \in A_{G,G^*}$ and $G^* \notin A_{G,G^*}$.
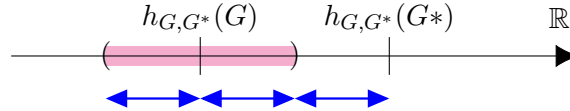

Figure 4: An illustration to better understand the proof. The set A_{G,G^*} consists of all graphs G that are mapped by h_{G,G^*} into the pink area, which size depends on the distance between the image of G and G^* under h_{G,G^*}. Moreover, the blue distances all have the same length.

Furthermore, by assumption h_{G,G^*} is continuous, such that A_{G,G^*} is an open set. For every $G \in \mathcal{X}$ with $G \simeq_{\text{1WL}} G^*$ we define A_{G,G^*} as follows:

$$A_{G,G^*} := \{G' \in \mathcal{X} \mid \|G' - G\|_2 < \epsilon\},$$

where $\|\cdot\|_2$ is the l_2.

Thus, $\{A_{G,G^*}\}_{G \in \mathcal{X}}$ is an open cover of $\mathcal{X}$. Since $\mathcal{X}$ is compact, there exists a finite subset $\mathcal{X}_0$ such that $\{A_{G,G^*}\}_{G \in \mathcal{X}_0}$ also covers $\mathcal{X}$. Hence, $\forall G \in \mathcal{X} \exists G_0 \in \mathcal{X}_0 : G \in A_{G_0,G^*}$.

We define the desired function h_{G^*} as follows:

$$h_{G^*}(\cdot) = \sum_{\substack{G_0 \in \mathcal{X}_0 \\ G_0 \not\simeq_{\mathrm{1WL}} G^*}} \overline{h}_{G_0,G^*}(\cdot), \tag{0.3}$$

where we define $\overline{h}_{G_0,G^*}$ almost exactly the same as in a previous proof:

$$\overline{h}_{G_0,G^*}(\cdot) = |h_{G_0,G^*}(\cdot) - h_{G_0,G^*}(G^*)| \tag{0.4}$$
$$= \max(h_{G_0,G^*}(\cdot) - h_{G_0,G^*}(G^*)) + \max(h_{G_0,G^*}(G^*) - h_{G_0,G^*}(\cdot)) \tag{0.5}$$

Note that, "$h_{G_0,G^*}(G^*)$" is a constant in the definition of $\overline{h}_{G_0,G^*}(\cdot)$. We will shortly proof, that $h_{G^*}(\cdot)$ fulfills the desired properties on input $G \in \mathcal{X}$:

1. By construction, any $\overline{h}_{G_0,G^*}$ is non-negative, such that the sum over these functions is also non-negative.

2. If $G \simeq_{\mathrm{1WL}} G^*$, using Lemma 19 we know that $h_{G^*}(G) = h_{G^*}(G^*)$, and by definition $h_{G^*}(G^*) = 0$, such that we can conclude $h_{G^*}(G) = 0$.

3. Let δ_{G^*} be:
$$\delta_{G^*} := \frac{1}{2} \min_{\substack{G_0 \in \mathcal{X} \\ G_0 \not\simeq_{\mathrm{1WL}} G^*}} |h_{G_0,G^*}(G_0) - h_{G_0,G^*}(G^*)|.$$

Prove by contraposition: Assume that for every graph $G' \in \mathcal{X}/\simeq_{\mathrm{1WL}}(G): \|G' - G^*\|_2 \geq \epsilon$. Hence, $G \notin \bigcup_{G' \in \mathcal{X}/\simeq_{\mathrm{1WL}}(G^*)} A_{G',G^*}$ (not in the union of l_2 balls of size ϵ around all graphs of the equivalence class of G^*). However, since $\{A_{G,G^*}\}_{G \in \mathcal{X}_0}$ is a cover of $\mathcal{X}$, there must exist a $G_0 \in \mathcal{X}_0$ with $G_0 \not\simeq_{\mathrm{1WL}} G^*$ such that $G \in A_{G_0,G^*}$. Thus, by definition of A_{G_0,G^*} we know that $h_{G_0,G^*}(G) \in (h_{G_0,G^*}(G_0) \pm \frac{|h_{G_0,G^*}(G_0) - h_{G_0,G^*}(G^*)|}{2})$, which when reformulated states:

$$|h_{G_0,G^*}(G) - h_{G_0,G^*}(G_0)| < \frac{1}{2}|h_{G_0,G_0}(G) - h_{G_0,G^*}(G^*)|. \tag{0.6}$$

Using this, we can prove $\overline{h}_{G_0,G^*}(G) \geq \delta_{G^*}$, which implies $h_{G^*}(G) \geq \delta_{G^*}$ and concludes the proof:

$$\overline{h}_{G_0,G^*}(G) = |h_{G_0,G^*}(G) - h_{G_0,G^*}(G^*)|$$
$$\geq |h_{G_0,G^*}(G_0) - h_{G_0,G^*}(G^*)| - |h_{G_0,G^*}(G) - h_{G_0,G^*}(G_0)| \tag{0.7}$$
$$\geq \frac{1}{2}|h_{G_0,G^*}(G_0) - h_{G_0,G^*}(G^*)| \tag{0.8}$$
$$\geq \frac{1}{2} \min_{\substack{G_0 \in \mathcal{X} \\ G_0 \not\simeq_{\mathrm{1WL}} G^*}} |h_{G_0,G^*}(G_0) - h_{G_0,G^*}(G^*)| =: \delta_{G^*}$$

To understand these inequalities, it helps to visualize them, see therefore Figure 5. We will try to give an explanation now, using the colored distances depicted in Figure 5. In Equation 0.7, we use the fact that the red distance is always greater than the green minus the blue distance. Further, using Equation 0.6, we know that the blue distance is always smaller than half of than the green distance. Using this fact, it is easy to see that in Equation 0.8 the green minus the blue distance is always greater than or equal to the half of the green distance.
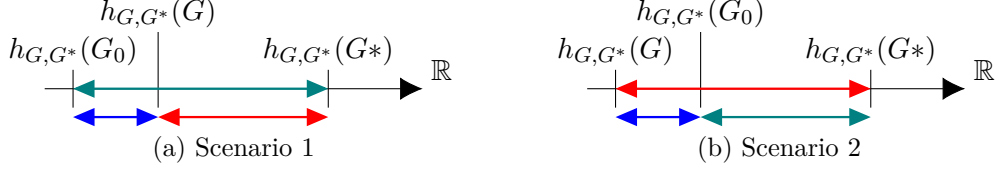
Figure 5: An illustration of two general scenarios to better understand the above used inequalities. Note that, there exists two more general scenarios, with $h_{G_0,G^*}(G^*) < h_{G_0,G^*}(G_0)$, but since we use the absolute function as our measure of distance, these scenarios are equivalent to the ones depicted due to the symmetry of the absolute function. In Table 1 below, we listed all colors used to decode the distances in the figures with their corresponding term in the inequalities.

Color	Term
red	$|h_{G_0,G^*}(G) - h_{G_0,G^*}(G^*)|$
green	$|h_{G_0,G^*}(G_0) - h_{G_0,G^*}(G^*)|$
blue	$|h_{G_0,G^*}(G) - h_{G_0,G^*}(G_0)|$

Table 1: Color legend for the proof of Lemma 26

Consequently, we know that if $h_{G^*}(G) < \delta_G$ it means that $G \in \bigcup_{G' \in \mathcal{X}/\simeq_{1\mathrm{WL}}(G^*)} A_{G',G^*}$, which implies that there exists $G' \in \mathcal{X}/\simeq_{1\mathrm{WL}}(G)$ with $\|G' - G^*\|_2 < \epsilon$.

As a final note, we can construct a multilayer perceptron $\mathrm{MLP}_{h_{G^*}}$ computing the function $h_{G^*}(\cdot)$ as in Equation 0.3. The $\mathrm{MLP}_{h_{G^*}}$ gets as input a vector of the output of the finite set of functions $\{\bar{h}_{G_0,G^*}\}_{G_0 \in \mathcal{X}_0, \ G_0 \not\simeq_{1\mathrm{WL}} G^*}$ applied on the input graph. Further, Equation 0.3 can be encoded by replacing the max operator by the non-linear activation function ReLU. Using Lemma 21, we can conclude that $h_{G^*}(\cdot)$ is computable by 1-WL+NN.

$\square$

Lemma 27. Let $\mathcal{C}$ be a collection of continuous functions from $\mathcal{X}$ to $\mathbb{R}$. If $\mathcal{C}$ is able to locate every $\simeq_{1\mathrm{WL}}$ equivalence class on $\mathcal{X}$, then there exists a collection of functions $\mathcal{C}'$ computable by 1-WL+NN that is able to locate every $\simeq_{1\mathrm{WL}}$ equivalence class on $\mathcal{X}$.

Proof. Let $\mathcal{A}$ be a continuous function from $\mathcal{X}$ to $\mathbb{R}$ computable by a GNN. Since $\mathcal{X}$ is compact, this implies that $\mathcal{A}$ is uniformaly continuous on $\mathcal{X}$, which further implies that $\forall \epsilon > 0 \exists r > 0$ such that $\forall G_1, G_2 \in \mathcal{X}$, if $\|G_1 - G_2\|_2 < r$, then $|\mathcal{A}(G_1) - \mathcal{A}(G_2)| < \epsilon$. Further, since $\mathcal{A}$ is GNN computable, we know that $\forall G_1, G_2 \in \mathcal{X}$, if $G_1 \simeq_{1\mathrm{WL}} G_2 : \mathcal{A}(G_1) = \mathcal{A}(G_2)$, hence if $G' \in \mathcal{X}/\simeq_{1\mathrm{WL}}(G_1)$ with $\|G' - G_2\|_2 < r$ exists, than $|\mathcal{A}(G_1) - \mathcal{A}(G_2)| < \epsilon$.

Throughout the rest of the proof, let $\epsilon > 0$ be fixed. Using the property described above, we know that for this ϵ there exists a constant r, we fix this aswell. Further, by the assumptions of the lemma we try to prove, we know that for any $\epsilon' > 0$ there exists $h_G \in \mathcal{C}$ for any $G \in \mathcal{X}$ with the above described properties. We choose $\epsilon' := r$ for all $h_G \in \mathcal{C}$ throughout the proof.

For any $G \in \mathcal{X}$, we define the set $h_G^{-1}(a) := \{G' \in \mathcal{X} \mid h_G(G') \in [0, a)\}$, as the set of graphs that are mapped into the open interval $[0, a)$ by h_G. We illustrated this set in Figure 6 for better understanding.
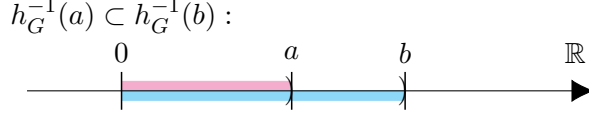
$$h_G^{-1}(a) \subset h_G^{-1}(b):$$

Figure 6: Illustration of the set $h_G^{-1}(\cdot)$ for two arbitrary values a, b with $a < b$. The pink area visualizes the open interval $[0, a)$, and similarly in blue for $[0, b)$.

Then, by definition of h_G, there exists a constant δ_G such that:

$$h_G^{-1}(\delta_G) \subseteq \bigcup_{G' \in \mathcal{X}/\simeq_{1\mathrm{WL}}(G)} \{G'' \in \mathcal{X} \mid \|G'' - G'\|_2 < r\}.$$

Since h_G is continuous by definition, $h_G^{-1}(\delta_G)$ is an open set. Hence, $\{h_G^{-1}(\delta_G)\}_{G \in \mathcal{X}}$ is an open cover of $\mathcal{X}$, and further as $\mathcal{X}$ is compact, there exists a finite subset $\mathcal{X}_0 \subseteq \mathcal{X}$ such that $\{h_{G_0}^{-1}(\delta_{G_0})\}_{G_0 \in \mathcal{X}_0}$ is also a cover of $\mathcal{X}$. For each $G_0 \in \mathcal{X}_0$, we construct the function φ_{G_0} from $\mathcal{X}$ to $\mathbb{R}$ as follows:

$$\varphi_{G_0}(\cdot) := \max(\delta_{G_0} - h_{G_0}(\cdot),\ 0).$$

The function has two important properties, for once it is non-negative and for any $G \in \mathcal{X}$: $\varphi_{G_0}(G) > 0$, if and only if $G \in h_{G_0}^{-1}(\delta_{G_0})$, thereby acting as a sort of weak indicator function. Building up on this property, we construct for each $G_0 \in \mathcal{X}_0$ the function ψ_{G_0} from $\mathcal{X}$ to $\mathbb{R}$ as follows:

$$\psi_{G_0}(\cdot) := \frac{\varphi_{G_0(\cdot)}}{\sum_{G' \in G_0} \varphi_{G'}(\cdot)},$$

which is well-defined, because $\{h_{G_0}^{-1}(\delta_{G_0})\}_{G_0 \in \mathcal{X}_0}$ is a cover of $\mathcal{X}$, such that we can conclude that for any input graph G on $\psi_{G_0}(\cdot)$ there exists a set $h_{G_0}^{-1}(\delta_{G_0})$ with $G \in h_{G_0}^{-1}(\delta_{G_0})$ implying $\varphi_{G_0}(G) > 0$, thus making the denominator not 0. The function ψ_{G_0} has two important properties, for once it is non-negative, because φ_G for all $G \in \mathcal{X}$ is non-negative, and for any $G \in \mathcal{X} : \psi_{G_0}(G) > 0$, if and only if $G \in h_{G_0}^{-1}(\delta_{G_0})$.

Further, we can observe that the set of functions $\{\psi_{G_0}\}_{G_0 \in \mathcal{X}_0}$ is a partition of unity on $\mathcal{X}$ with respect to the open cover $\{h_{G_0}^{-1}(\delta_{G_0})\}_{G_0 \in \mathcal{X}_0}$, because:

1. For any $G \in \mathcal{X}$ the set of functions mapping G not to 0 is finite, as the set of all functions $\{\psi_{G_0}\}_{G_0 \in \mathcal{X}_0}$ is finite, since $\mathcal{X}_0$ is finite.

2. For any $G \in \mathcal{X} : \sum_{G_0 \in \mathcal{X}_0} \psi_{G_0}(G) = 1$, since:

$$\sum_{G_0 \in \mathcal{X}_0} \psi_{G_0}(G) = \sum_{G_0 \in \mathcal{X}_0} \frac{\varphi_{G_0(G)}}{\sum_{G' \in \mathcal{X}_0} \varphi_{G'}(G)} = \frac{\sum_{G_0 \in \mathcal{X}_0} \varphi_{G_0(G)}}{\sum_{G' \in \mathcal{X}_0} \varphi_{G'}(G)} = 1.$$

We can use this property to decompose the given function $\mathcal{A}$ as follows on any input $G \in \mathcal{X}$:

$$\mathcal{A}(G) = \mathcal{A}(G) \cdot \Big(\sum_{G_0 \in \mathcal{X}_0} \psi_{G_0}(G) \Big) = \sum_{G_0 \in \mathcal{X}_0} \mathcal{A}(G) \cdot \psi_{G_0}(G).$$

Recall the property from the beginning of the proof that for any $G \in \mathcal{X}$ if $G \in h_{G_0}^{-1}(\delta_{G_0})$, then there exists a $G' \in \mathcal{X}/\simeq_{1\mathrm{WL}}(G)$ with $\|G' - G_0\|_2 < r$, which implies that $|\mathcal{A}(G) - \mathcal{A}(G_0)| < \epsilon$. With this, we can construct a function $\hat{\mathcal{A}}$ on $\mathcal{X}$ that approximates $\mathcal{A}$ within accuracy ϵ:

$$\hat{\mathcal{A}}(\cdot) = \sum_{G_0 \in \mathcal{X}_0} \mathcal{A}(G_0) \cdot \psi_{G_0}(\cdot).$$

18

Note that, "$\mathcal{A}(G_0)$" is a constant here. To prove that $\hat{\mathcal{A}}$ approximates $\mathcal{A}$ within accuracy ϵ we need to show that $\sup_{G \in \mathcal{X}} |\mathcal{A}(G) - \hat{\mathcal{A}}(G)| < \epsilon$. Let $G \in \mathcal{X}$ be arbitrary, then:

$$
\begin{aligned}
|\mathcal{A}(G) - \hat{\mathcal{A}}(G)| &= \Big| \sum_{G_0 \in \mathcal{X}_0} \mathcal{A}(G) \cdot \psi_{G_0}(G) \ - \sum_{G_0 \in \mathcal{X}_0} \mathcal{A}(G_0) \cdot \psi_{G_0}(G) \Big| \\
&= \sum_{G_0 \in \mathcal{X}_0} |\mathcal{A}(G) - \mathcal{A}(G_0)| \cdot \psi_{G_0}(G) \\
&< \sum_{G_0 \in \mathcal{X}_0} \epsilon \cdot \psi_{G_0}(G) \\
&= \epsilon \cdot \sum_{G_0 \in \mathcal{X}_0} \psi_{G_0}(G) \\
&= \epsilon \cdot 1
\end{aligned}
\tag{0.9}
$$

In Equation 0.9, we use the fact that applies to all $G_0 \in \mathcal{X}_0$ on input G:

- If $\psi_{G_0}(G) > 0$, than we know that $G \in h_{G_0}^{-1}(\delta_{G_0})$, such that we can upper bound $|\mathcal{A}(G) - \mathcal{A}(G_0)| < \epsilon$.

- If $\psi_{G_0}(G) = 0$, we know that the no matter what $|\mathcal{A}(G) - \mathcal{A}(G_0)|$ is, the summand is 0, such that we can just assume $|\mathcal{A}(G) - \mathcal{A}(G_0)| < \epsilon$ without loss of generality.

In the end, we give a short explanation, that $\hat{\mathcal{A}}$ is computable by 1-WL+NN. For this we construct a multilayer perceptron with three layers and then conclude with Lemma 21 the computability. Let us therefore break down the whole construction of $\hat{\mathcal{A}}$:

$$
\hat{\mathcal{A}}(\cdot) = \sum_{G_0 \in \mathcal{X}_0} \mathcal{A}(G_0) \cdot \frac{1}{\sum_{G' \in \mathcal{X}_0} \max(\delta_{G'} - h_{G'}(\cdot),\, 0)} \cdot \max(\delta_{G_0} - h_{G_0}(\cdot),\, 0).
$$

We construct a multilayer perceptron $\mathrm{MLP}_{\hat{\mathcal{A}}}$ that takes in as input a vector of the output of the finite set of functions $\{h_{G_0}\}_{G_0 \in \mathcal{X}_0}$ applied on the input graph. In the first layer we compute each $\max(\delta_{G_0} - h_{G_0}(\cdot))$ term, where δ_{G_0} is a constant, in particular the bias, and the max operator is replaced by the activation function ReLU. In the second layer, we compute the sum of the denominator of the fraction, to which we apply the activation function $f(x) := \frac{1}{x}$. In the last layer we compute the overall sum where $\mathcal{A}(G_0)$ is a constant. $\qquad \square$

4.4 Proof of Theorem 16

In this section we will shortly prove that GNN-Approximating implies 1-WL-Discriminating. Similar to the proof of Theorem 12, we will take advantage of the fact that our Definition 7 of GNNs does not impose any constraints on the readout function, apart from the fact that it must be permutation invariant and computable. We have drawn inspiration for this proof from the work of Chen et al. [2019].

Proof of Theorem 16. Let $\mathcal{C}$ be a collection of continues functions from $\mathcal{X}$ to $\mathbb{R}$ that is GNN approximating. Let $G_1, G_2 \in \mathcal{X}$ with $G_1 \not\simeq_{\mathrm{1WL}} G_2$, then we define the function h_{G_1,G_2} on input $G \in \mathcal{X}$:

$$
h_{G_1,G_2}(G) = \min_{\pi \in S_n} \|\pi^T G \pi - G_1\|_2,
$$

where $\|\cdot\|_2$ is the l_2 norm. Note that, h_{G_1,G_2} is computable as the number of permutations π in S_n are finite. Since h_{G_1,G_2} is permutation invariant, we can construct a GNN with 0 layers

19

and h_{G_1,G_2} as its READOUT function, thereby constructing a GNN computing h_{G_1,G_2} and consequently showing that the function is GNN computable. We choose $\epsilon := \frac{1}{2} \cdot h_{G_1,G_2}(G_2) > 0$, then there exists a function $h_\epsilon \in \mathcal{C}$ approximating h_{G_1,G_2} within ϵ accuracy. With this, we have a function $h_\epsilon \in \mathcal{C}$ that can distinguish G_1 and G_2, since:

$$
\begin{aligned}
|h_\epsilon(G_1) - h_\epsilon(G_2)| &> |(h_{G_1,G_2}(G_1) - \epsilon) - (h_{G_1,G_2}(G_2) + \epsilon)| \\
&= |h_{G_1,G_2}(G_2) - 2 \cdot \epsilon| \qquad\qquad \text{by definition } h_{G_1,G_2}(G_1) = 0 \\
&= |h_{G_1,G_2}(G_2) - 2 \cdot \frac{1}{2} \cdot h_{G_1,G_2}(G_2)| = 0.
\end{aligned}
$$

$\square$

5 Empirical Investigation

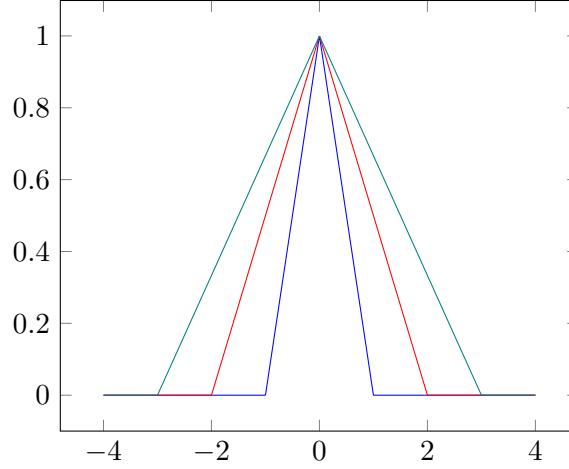Yet to come!

6 Appendix

6.1 Figures and graphs

Figure 7: Illustration of the so-called "bump" function $\psi_a(x)$ used in the proof of Lemma 23. Here the colors of the displayed functions correspond to the parameter a set to $a := 1$ in blue, $a := 2$ in red and $a := 3$ in green.

6.2 Proofs

Proof of Lemma 21. Let $\mathcal{C}$ be a collection of functions computed by 1-WL+NN, $h_1, \ldots, h_n \in \mathcal{C}$, and MLP$^\bullet$ a multilayer perceptron. Further, let $f_1, \ldots, f_n$ be the encoding functions, as well as $\mathrm{MLP}_1, \ldots, \mathrm{MLP}_n$ be the multilayer perceptrons used by $h_1, \ldots h_n$ respectively. As outlined above, we will now construct f^* and MLP*, such that for all graphs $G \in \mathcal{X}$:

$$\mathrm{MLP}^\bullet(h_1(G), \ldots, h_n(G)) = \mathrm{MLP}^* \circ f^* \circ \text{1-WL}(G)$$

with which we can conclude that the composition of multiple functions computable by 1-WL+NN, is in fact also 1-WL+NN computable.

We define the new encoding function f^* to work as follows on input C_∞:

$$f^*(C_\infty) := \mathsf{concat}(\begin{bmatrix} f_1(C_\infty) \\ \vdots \\ f_n(C_\infty) \end{bmatrix}),$$

where $\mathsf{conccat}$ is the concatenation function, concatenating all encoding vectors to one single vector.

Using the decomposition introduced in Definition 8, we can decompose each MLP_i for $i \in [n]$ at layer $j > 0$ as follows: $(\mathrm{MLP}_i)_j(v) := \sigma(W_j^i \cdot (\mathrm{MLP}_i)_{j-1}(v) + b_j^i)$. Using this notation we

construct MLP* as follows:

$$(\text{MLP}^*)_0(v) := v$$

$$(\text{MLP}^*)_{j+1}(v) := \sigma(W_j^* \cdot (\text{MLP}^*)_j(v) + \mathsf{concact}(\begin{bmatrix} b_j^1 \\ \vdots \\ b_j^n \end{bmatrix})) \qquad\qquad , \forall j \in [k]$$

$$(\text{MLP}^*)_{j+k+1}(v) := (\text{MLP}^\bullet)_{j+1}(v) \qquad\qquad , \forall j \in [k^\bullet - 1]$$

where k is the maximum number of layers of the set of MLP_i's, $k^\bullet$ is the number of layers of the given $\text{MLP}^\bullet$ and σ an element wise activation function. Thereby, we define in the first equation line, that the start of the sequence is the input, with the second line, we construct the "simultaneous" execution of the MLP_i's, and in the last equation line, we add the layers of the given $\text{MLP}^\bullet$ to the end. Further, we define the weight matrix W_j^* as follows:

$$W_j^* := \begin{bmatrix} W_j^1 & 0 & \dots & 0 \\ 0 & W_j^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & W_j^n \end{bmatrix},$$

such that we build a new matrix where each individual weight matrix is placed along the diagonal. Here we denote with 0, zero matrices with the correct dimensions, such that W_j^* is well-defined. Important to note, should for an MLP_i, W_j^i not exist, because it has less than j layers, we use for W_j^i the identity matrix I_m where m is the dimension of the output computed by MLP_i. $\qquad\square$

Bibliography

[1] J. Cai, M. Fürer, and N. Immerman. An optimal lower bound on the number of variables for graph identifications. *Combinatorica*, 12(4):389–410, 1992.

[2] A. Cardon and M. Crochemore. Partitioning a graph in $O(|A| \log_2 |V|)$. *Theoretical Computer Science*, 19(1):85 – 98, 1982.

[3] Z. Chen, S. Villar, L. Chen, and J. Bruna. On the equivalence between graph isomorphism testing and function approximation with GNNs. In *Advances in Neural Information Processing Systems*, pages 15868–15876, 2019.

[4] M. Grohe. *Descriptive Complexity, Canonisation, and Definable Graph Structure Theory.* Lecture Notes in Logic. Cambridge University Press, 2017.

[5] W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1025–1035, 2017.

[6] C. Morris, N. M. Kriege, K. Kersting, and P. Mutzel. Faster kernel for graphs with continuous attributes via hashing. In *IEEE International Conference on Data Mining*, pages 1095–1100, 2016.

[7] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe. Weisfeiler and Leman go neural: Higher-order graph neural networks. In *AAAI Conference on Artificial Intelligence*, pages 4602–4609, 2019.

[8] R. Paige and R. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.

[9] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.