4. Theoretical Connection

This section forms the core of our theoretical investigation, where we explore the equivalence between two frameworks: 1-WL+NN and GNN. We present two theorems to establish this equivalence, each showing a separate equivalence direction. By combining these theorems, we conclusively establish the equivalence. To maintain clarity and rigor, we will prove each theorem separately afterward in a corresponding subsection.

In particular, the theorem will establish a theoretical connection between the frameworks when applied to a finite collection of graphs, which we denote by $\mathcal{X}$ with $\mathcal{X} \subset \mathcal{G}$.

Theorem 11 (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 12 (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 two theorems, the equivalence between both frameworks follows. Specifically, every function computed by 1-WL+NN working over any arbitrary but finite $\mathcal{X} \subset \mathcal{G}$ is also computable by a GNN, and vice versa. Consequently, we can draw the following corollary:

Corollary 13. For an arbitrary function f working over $\mathcal{X}$ to $\mathbb{R}$: The function f is computable by a 1-WL+NN model, if and only if, the function f is computable by a GNN model.

As we move towards the empirical evaluation in Part II, it is evident that if we test a 1-WL+NN model on any of the benchmark datasets, it can theoretically achieve the same level of performance as a GNN model. Notice that we did not leverage any constraints on the encoding of graphs throughout the two theorems and their corresponding proves but instead kept it general.

4.1. Proof of Theorem 11: "GNN $\subseteq$ 1-WL+NN"

We will prove Theorem 12 by first introducing a set of lemmas, which will be leveraged in the proof of the theorem at the end of this section. The lemmas Lemmas 18 and 19, and the proof of Theorem 11 extend the results by Chen et al. [2019].

To begin, we prove an essential insight that for any pair of graphs indistinguishable by the 1-WL isomorphism test, the output of any 1-WL+NN model applied to both graphs is identical.

Lemma 14 (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 $G_1 \simeq_{1\text{WL}} G_2$ then $\mathcal{B}(G_1) = \mathcal{B}(G_2)$.

Proof of Lemma 14. Assume the above. Let $\mathcal{B}$ be an arbitrary function over $\mathcal{X}$ computable by 1-WL+NN, then $\mathcal{B}$ is composed as follows: $\mathcal{B}(\cdot) = \mathsf{MLP} \circ f_{\text{enc}}\{\!\{\text{1-WL}(\cdot)(v) \mid v \in V(\cdot)\}\!\}$. Further, let $G_1, G_2 \in \mathcal{X}$ be arbitrary graphs with $G_1 \simeq_{1\text{WL}} G_2$, then by definition of the $\simeq_{1\text{WL}}$ relation we know that $\text{1-WL}(G_1) = \text{1-WL}(G_2)$. With this, the equivalence follows, since this implies the equivalence of the multiset of colors for both graphs. □

As a consequence of this lemma, we establish that every function computable by 1-WL+NN is also permutation invariant.

Corollary 15 (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 of Corollary 15. Assuming the above, let $G \in \mathcal{X}$ be an arbitrary graph, and π be a permutation of $V(G)$, the set of nodes. By the definition of isomorphism, we know that G is isomorph to $\pi \cdot G$. Further, since the 1-WL isomorphism test is sound, we know that $G \simeq \pi \cdot G$ implies $G \simeq_{1\text{WL}} \pi \cdot G$. Using Lemma 14, we can therefore conclude that: $\mathcal{B}(G) = \mathcal{B}(\pi \cdot G)$. □

With this property of 1-WL+NN functions, we can show the existence of a 1-WL-Discriminating collection of functions computable by 1-WL+NN with Lemma 16. It is necessary to prove this lemma as it forms the basis of Lemma 18.

Lemma 16. 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 of Lemma 16. We will prove the lemma by giving a construction of such a collection. In particular, we define the collection $\mathcal{C}$ as follows:

$$\mathcal{C} := \{\mathcal{B}_c : \mathcal{X} \to \mathbb{R},\ G \mapsto \mathsf{MLP}_{\text{id}} \circ f_c(\{\!\{\text{1-WL}(G)(v) \mid v \in V(G)\}\!\}) \mid c \in \mathbb{N}\},$$

where MLP_{id} is the identity function encoded as a multilayer perceptron that returns its input and f_c is an encoding function that returns the number of nodes colored as c. Since every function $\mathcal{B}_c \in \mathcal{C}$ is composed of the 1-WL algorithm, an encoding function f_c, and a multilayer perceptron MLP_{id}, each function is computable by 1-WL+NN, and consequently, also the whole collection.

To prove that this collection is 1-WL-Discriminating, we need to show two properties: 1) Each function in the collection is permutation invariant, and 2) For each pair of graphs in $\mathcal{X}$ distinguishable by the 1-WL isomorphism test, there must exist a function in the collection that also distinguishes the pair.

For the first property, we already established in Corollary 15 that all 1-WL+NN functions are permutation invariant. For the second property, let $G_1, G_2 \in \mathcal{X}$ with $G_1 \not\simeq_{1\text{WL}} G_2$. Further, 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 $\text{hist}_{G_1,C_1}(c) \neq \text{hist}_{G_2,C_2}(c)$, such that $\mathcal{B}_c \in \mathcal{C}$ exists with $\mathcal{B}_c(G_1) \neq \mathcal{B}_c(G_2)$, satisfying the second property. □

The following Lemma 17 forms the basis in constructing 1-WL+NN computable functions in the subsequent proofs of the Lemmas 18 and 19, as well as in the proof of the actual theorem. Specifically, it shows that combining the output of multiple 1-WL+NN computable functions and processing them further with a multilayer perceptron is also 1-WL+NN computable.

Lemma 17 (1-WL+NN Composition)**.** Let $\mathcal{C}$ be a collection of functions computable by 1-WL+NN. Further, let $h_1, \dots h_n \in \mathcal{C}$ and $\mathsf{MLP}^\bullet$ be a multilayer perceptron operating from $\mathbb{R}^n$ to $\mathbb{R}$, then the function $\mathcal{B}$ composed as follows:

$$\mathcal{B} : \mathcal{X} \to \mathbb{R},\ G \mapsto \mathsf{MLP}^\bullet\left(\begin{bmatrix} h_1(G) \\ \vdots \\ h_n(G) \end{bmatrix}\right),$$

is also computable by 1-WL+NN.

Proof of Lemma 17. Assume the above. Let $f_1, \ldots, f_n$ be the encoding functions, and $\mathsf{MLP}_1, \ldots, \mathsf{MLP}_n$ be the multilayer perceptrons used by $h_1, \ldots, h_n$, respectively. The key idea of this proof is to construct an encoding function f^* that "duplicates" its input and applies each encoding function f_i followed by each multilayer perceptron MLP_i individually. Afterward, we apply $\mathsf{MLP}^\bullet$ to the resulting concatenated vector. Thus, we can represent $\mathcal{B}$ as follows:

$$\mathcal{B}(\cdot) = \mathsf{MLP}^\bullet \circ f^*(\{\!\{\mathsf{1\text{-}WL}(\cdot)(v) \mid v \in V(\cdot)\}\!\}).$$

In detail, we define the encoding function f^* as follows:

$$f^*(\cdot) := \mathsf{concat}(\begin{bmatrix} \mathsf{MLP}_1 \circ f_1(\cdot) \\ \vdots \\ \mathsf{MLP}_n \circ f_n(\cdot) \end{bmatrix}),$$

where concat is the concatenation function that combines all encoding vectors into a single vector. By doing so, we have shown that $\mathcal{B}$ can be decomposed into three components of a $\mathsf{1\text{-}WL{+}NN}$ model, allowing us to conclude that it is $\mathsf{1\text{-}WL{+}NN}$ computable. □

In the following two Lemmas 18 and 19, we will show that the indicator function $\mathbb{1}$ for the $\simeq_{1\mathrm{WL}}$ relation on $\mathcal{X}$ is $\mathsf{1\text{-}WL{+}NN}$ computable. We formally define this function for any pair of graphs $G_1, G_2 \in \mathcal{X}$ as follows:

$$\mathbb{1}_{G_1 \simeq_{1\mathrm{WL}} G_2} = \begin{cases} 1, & \text{if } G_1 \simeq_{1\mathrm{WL}} G_2 \\ 0, & \text{else} \end{cases}.$$

This function plays a crucial role in the proof of the theorem. We will first introduce an approximation of the inverse of this function in Lemma 18 and then use this approximation for the proof of Lemma 19 to construct a function $\varphi_{G_1}(G_2)$ that is equivalent to the indicator function $\mathbb{1}_{G_1 \simeq_{1\mathrm{WL}} G_2}$.

Lemma 18. Let $\mathcal{C}$ be a collection of functions from $\mathcal{X}$ to $\mathbb{R}$ computable by $\mathsf{1\text{-}WL{+}NN}$ that is $\mathsf{1\text{-}WL\text{-}Discriminating}$. Then for all $G^* \in \mathcal{X}$, there exists a function h_{G^*} from $\mathcal{X}$ to $\mathbb{R}$ computable by $\mathsf{1\text{-}WL{+}NN}$, such that on any input $G \in \mathcal{X} : h_{G^*}(G) = 0$, if and only if, $G \simeq_{1\mathrm{WL}} G^*$.

Proof of Lemma 18. Assume the above. Since $\mathcal{C}$ is $\mathsf{1\text{-}WL\text{-}Discriminating}$, we know that for any pair of graphs $G_1, G_2 \in \mathcal{X}$ with $G_1 \not\simeq_{1\mathrm{WL}} G_2$, the function $h_{G_1,G_2} \in \mathcal{C}$ exists, that distinguishes the pair, such that $h_{G_1,G_2}(G_1) \neq h_{G_1,G_2}(G_2)$. We define the function $\overline{h}_{G_1,G_2}$ working over $\mathcal{X}$ for every such pair as follows:

$$\begin{aligned} \overline{h}_{G_1,G_2}(\cdot) &= |h_{G_1,G_2}(\cdot) - h_{G_1,G_2}(G_1)| \\ &= \max(h_{G_1,G_2}(\cdot) - h_{G_1,G_2}(G_1),\ 0) + \max(h_{G_1,G_2}(G_1) - h_{G_1,G_2}(\cdot),\ 0) \\ &= \mathrm{ReLU}(h_{G_1,G_2}(\cdot) - h_{G_1,G_2}(G_1)) + \mathrm{ReLU}(h_{G_1,G_2}(G_1) - h_{G_1,G_2}(\cdot)) \end{aligned} \tag{4.1}$$

Note, that in the equations 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, then we will construct the function h_{G_1} with the desired properties as follows:

$$h_{G_1}(\cdot) = \sum_{\substack{G_2 \in \mathcal{X} \\ G_1 \not\simeq_{1\mathrm{WL}} G_2}} \overline{h}_{G_1,G_2}(\cdot). \tag{4.2}$$

Since $\mathcal{X}$ is finite, the sum is finite and therefore well-defined. Next, we will show that this construction fulfills the desired properties, by proving that for any input $G \in \mathcal{X} : h_{G_1}(G) = 0$, if and only if, $G \simeq_{1\text{WL}} G_1$. Note that G_1 is arbitrary but fixed. Let $G \in \mathcal{X}$ be an arbitrary input graph:

1. If $G_1 \simeq_{1\text{WL}} G$, then for every function $\overline{h}_{G_1,G_2}$ of the sum with $G_1 \not\simeq_{1\text{WL}} G_2$, we know, using Lemma 14, 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 $G_1 \not\simeq_{1\text{WL}} G$, 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 $\overline{h}_{G_1,G_2}$ function.

Using Lemma 17, we can conclude that for any $G \in \mathcal{X}$, h_G is computable by 1-WL+NN, as we can encode Equation (4.2) via a multilayer perceptron MLP where the constant $h_{G_1,G_2}(G_1)$ of Equation (4.1) will be the bias of the corresponding channel, such that the MLP exists.

It is crucial to note that in the special case where no pair of graphs within $\mathcal{X}$ is indistinguishable by the 1-WL isomorphism test from another, the function h_{G_1} is still defined. However, it sums over zero summands, resulting in $h_{G_1}(\cdot) = 0$. □

Thus, we proved that the function h_G is 1-WL+NN computable for any graph $G \in \mathcal{X}$. The function h_G approximates the inverted indicator function for the fixed graph G by mapping graphs indistinguishable from G by the 1-WL algorithm to 0 while mapping every other graph to something strictly larger than 0. The following proof will use this property to construct the indicator function.

Lemma 19. Let $\mathcal{C}$ be a collection of functions from $\mathcal{X}$ to $\mathbb{R}$ computable by 1-WL+NN such that for all $G^* \in \mathcal{X}$, there exists $h_{G^*} \in \mathcal{C}$ satisfying $h_{G^*}(G) = 0$, if and only if, $G \simeq_{1\text{WL}} G^*$, 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 $G \in \mathcal{X}$: $\varphi_{G^*}(G) = \mathbb{1}_{G \simeq_{1\text{WL}} G^*}$.

Proof of Lemma 19. Assume the above. Due to $\mathcal{X}$ being finite, we can define for every graph G^* the constant:

$$\delta_{G^*} \coloneqq \frac{1}{2} \min_{\substack{G \in \mathcal{X} \\ G \not\simeq_{1\text{WL}} G^*}} |h_{G^*}(G)|.$$

The constant δ_{G^*} represents the minimum value to which the corresponding function $h_{G^*}(\cdot)$ maps a graph G that is distinguishable from G^* by the 1-WL isomorphism test, multiplied by the factor $\frac{1}{2}$. The specific value of this factor is arbitrary; the crucial aspect is that it remains less than 1, ensuring that the constant δ_{G^*} remains strictly smaller than the minimum value of $h_{G^*}(\cdot)$ for any graph G where $G \not\simeq_{1\text{WL}} G^*$.

It is important to note that in the special case where no pair of graphs within $\mathcal{X}$ is indistinguishable by the 1-WL isomorphism test from another, the constant δ_{G^*} is not well-defined. For these cases, we set $\delta_{G^*} := 1$ for all $G^* \in \mathcal{X}$.

We further introduce a so-called "bump" function $\psi_a(x)$ working from $\mathbb{R}$ to $\mathbb{R}$ paraemetrized by $a \in \mathbb{R}$ with $a > 0$ and defined as follows:

$$\begin{aligned} \psi_a(x) &\coloneqq \max(\frac{x}{a} - 1,\ 0) + \max(\frac{x}{a} + 1,\ 0) - 2 \cdot \max(\frac{x}{a},\ 0) \\ &= \text{ReLU}(\frac{x}{a} - 1) + \text{ReLU}(\frac{x}{a} + 1) - 2 \cdot \text{ReLU}(\frac{x}{a}) \end{aligned} \tag{4.3}$$

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 5 for a plot of the relevant part of this function with exemplary values for a.

We use these properties and the constant δ_{G^*} to define for every graph $G^* \in \mathcal{X}$ the function φ_{G^*} that is equivalent to the indicator function as follows:

$$\varphi_{G^*}(\cdot) \coloneqq \psi_{\delta_{G^*}}(h_{G^*}(\cdot)).$$

We will prove the correctness of this construction by showing that for a fixed graph G^* the following condition holds: $\forall G \in \mathcal{X} : \varphi_{G^*}(G) = \mathbb{1}_{G \simeq_{\text{1WL}} G^*}$. For this, consider two cases:

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^*} using Equation (4.3) via a multilayer perceptron MLP, where δ_{G^*} is a constant, such that the MLP exists. With Lemma 17 we can therefore conclude that φ_{G^*} is computable by 1-WL+NN for every graph $G^* \in \mathcal{X}$. □

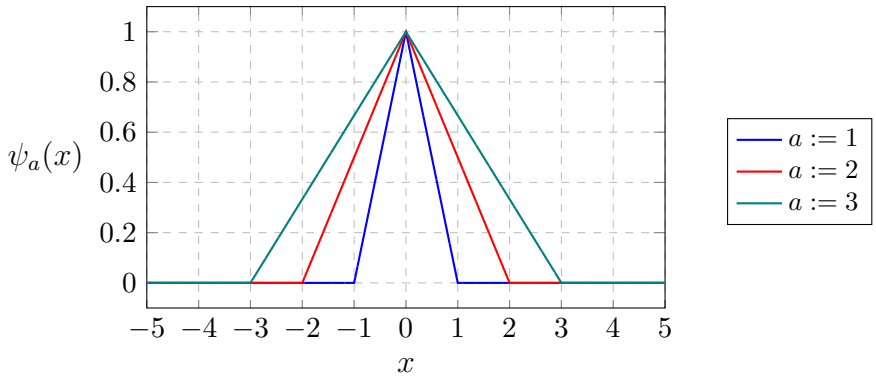

Figure 5.: Illustration of the so-called "bump" function $\psi_a(x)$ used in the proof of Lemma 19 with different exemplary values for a.

We can now leverage all our lemmas to prove the overall theorem that any function $\mathcal{A}$ computable by a GNN is also computable by 1-WL+NN.

Proof of Theorem 11. Let $\mathcal{A}$ be a function that works over $\mathcal{X}$ to $\mathbb{R}$ computed by a GNN model. We will prove that $\mathcal{A}$ is 1-WL+NN computable by decomposing the function and then argue that the decomposition is computable by a 1-WL+NN model. For this let $G \in \mathcal{X}$ be an arbitrary

input graph, we can decompose $\mathcal{A}(G)$ as follows:

$$\mathcal{A}(G) = \Big(\frac{1}{|\mathcal{X}/\simeq_{1\mathrm{WL}}(G)|} \sum_{G^* \in \mathcal{X}} \mathbb{1}_{G \simeq_{1\mathrm{WL}} G^*} \Big) \cdot \mathcal{A}(G) \tag{4.4}$$

$$= \sum_{G^* \in \mathcal{X}} \frac{1}{|\mathcal{X}/\simeq_{1\mathrm{WL}}(G)|} \cdot \mathcal{A}(G) \cdot \mathbb{1}_{G \simeq_{1\mathrm{WL}} G^*} \tag{4.5}$$

$$= \sum_{G^* \in \mathcal{X}} \frac{1}{|\mathcal{X}/\simeq_{1\mathrm{WL}}(G^*)|} \cdot \mathcal{A}(G^*) \cdot \mathbb{1}_{G \simeq_{1\mathrm{WL}} G^*} \tag{4.6}$$

$$= \sum_{G^* \in \mathcal{X}} \frac{\mathcal{A}(G^*)}{|\mathcal{X}/\simeq_{1\mathrm{WL}}(G^*)|} \cdot \varphi_{G^*}(G) \tag{4.7}$$

where $\mathcal{X}/\simeq_{1\mathrm{WL}}(G)$ denotes the set of all graphs that are equivalent to G according to the $\simeq_{1\mathrm{WL}}$ relation. We explain each equation step by step:

Equation (4.4): Multiplying $\mathcal{A}(G)$ by the factor in the parentheses is correct because it is equal to 1. This is because the sum "counts" the number of graphs $G^* \in \mathcal{X}$ that are indistinguishable from the input graph G by the 1-WL isomorphism test, and then the count is divided by the number of graphs in the equivalence class of G, which is the same as the count.

Equation (4.5): We can move both the factor $\mathcal{A}(G)$ and $\frac{1}{|\mathcal{X}/\simeq_{1\mathrm{WL}}(G)|}$ into the sum by using the distributive property of the space $\mathbb{R}$.

Equation (4.6): Due to the output of the indicator function $\mathbb{1}$ being either 0 or 1, we can infer that the inner product of each summand can only be nonzero if G^* is indistinguishable by the 1-WL isomorphism test from G. This implies that both are in the same equivalence class in these cases, such that $|\mathcal{X}/\simeq_{1\mathrm{WL}}(G)|$ is equal to $|\mathcal{X}/\simeq_{1\mathrm{WL}}(G^*)|$.
Additionally, since GNNs are, at most, as good as the 1-WL algorithm in distinguishing pairs of non-isomorphic graphs (Morris et al. [2019], Xu et al. [2019]), we can use the fact that for every graph $G^* \in \mathcal{X}$: if $G^* \simeq_{1\mathrm{WL}} G$, then $\mathcal{A}(G^*) = \mathcal{A}(G)$. Using the same reasoning with the indicator function, we can replace the term $\mathcal{A}(G)$ by $\mathcal{A}(G^*)$.

Equation (4.7): Utilizing Lemma 19, we can replace the indicator function with $\varphi_{G^*}(G)$.

In conclusion, we have decomposed the GNN function $\mathcal{A}(G)$ such that the only factors that depend on the input graph G are the functions φ_{G^*}, which take G as input. This observation implies that all other factors are constants. Consequently, we can reason that the entire decomposition can be computed by a multilayer perceptron with a single layer, which takes the output of all φ_{G^*} for all $G^* \in \mathcal{X}$, applied to the input graph G. The multilayer perceptron then multiplies each of these values with the constant $\frac{\mathcal{A}(G^*)}{|\mathcal{X}/\simeq_{1\mathrm{WL}}(G^*)|}$ and takes the sum. The existence of such a multilayer perceptron is evident, and when combined with Lemma 17, we can assert that this composition is 1-WL+NN computable.

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. □

4.2. Proof of Theorem 12: "1-WL+NN $\subseteq$ GNN"

In this section, we will prove the converse direction. Similar to the previous subsection, we will begin by introducing a set of lemmas that will play a crucial role in proving Theorem 12.

We start by showing the existence of a collection of functions computable by GNNs that is 1-WL-Discriminating. For the proof, we will devise message-passing layers for a GNN that effectively compute a single iteration of the 1-WL algorithm per layer. Afterward, we show that with a proper choice of the Readout function, we construct a collection of GNN functions that is 1-WL-Discriminating. Although prior works by Morris et al. [2019] and Xu et al. [2019] have already demonstrated how to construct message-passing layers to compute a single iteration of the 1-WL algorithm per layer, we include our own construction with our notation in the proof for two crucial reasons. Firstly, it ensures the completeness of our proof without assuming major parts. Secondly, and most importantly, it effectively highlights the remarkable similarities and key distinctions between the 1-WL algorithm and GNNs in general.

Lemma 20 (GNN 1-WL-Discriminating)**.** There exists a collection $\mathcal{C}$ of functions from $\mathcal{X}$ to $\mathbb{R}$ computable by GNNs that is 1-WL-Discriminating.

Proof of Lemma 20. Due to $\mathcal{X}$ being finite, we define the constants n, m, k as follows:

$$n := \max_{G \in \mathcal{X}} |V(G)|, \quad m := \sum_{G \in \mathcal{X}} |V(G)|, \quad \text{and} \quad k := 1 + \max_{\substack{G \in \mathcal{X} \\ v \in V(G)}} |l_G(v)|,$$

such that n is the maximum number of nodes of any graph in $\mathcal{X}$, m is the total number of nodes of the set $\mathcal{X}$, and k is the largest label of any node of a graph in $\mathcal{X}$ plus 1.

We will utilize these constants to construct the collection of functions $\mathcal{C} := \{\mathcal{A}_c \mid c \in \mathbb{N}\}$ that is 1-WL-Discriminating. For the remainder of this proof, we first describe the construction of an arbitrary $\mathcal{A}_c \in \mathcal{C}$ and afterward, prove that the collection $\mathcal{C}$ is 1-WL-Discriminating.

Each $\mathcal{A}_c$ consists of n message-passing layers. We define the input layer $f^{(0)}(v) := v$ as the identity functions such that there is no preprocessing of the node labels. Further, we define every other layer t with $1 \leq t \leq n$ as follows:

$$f^{(t)}(v) := f^{(t)}_{\text{merge}}(f^{(t-1)}(v),\ \{\!\{f^{(t-1)}(u) \mid u \in \mathcal{N}(v)\}\!\}).$$

Here $f^{(t)}_{\text{merge}}$ is an injective function that maps its input into its codomain:

$$\{i \in \mathbb{N} \mid k + (t-1) \cdot m \ \leq \ i \ \leq \ k + t \cdot m\}$$

This function exists due to the finiteness of $\mathcal{X}$. We can upper bound the cardinality of its domain, the number of unique tuples, by the total number of nodes in $\mathcal{X}$, which is m, and since the cardinality of its codomain is exactly m, we can conclude the existence of the function.

Next, we will define the Readout function of $\mathcal{A}_c$ to be the function that returns the number of nodes colored as c in the coloring of $f^{(n)}$.

By leveraging the results of *theorem 3* from the work of Xu et al. [2019], we can infer that each layer of each $\mathcal{A}_c$ computes a single iteration of the 1-WL algorithm. This observation makes sense as the update equation for each layer injectively maps each tuple to a previously unused color, similar to how the Relabel function of the 1-WL algorithm works. Moreover, since the 1-WL algorithm terminates on any graph $G \in \mathcal{X}$ after at most $|V(G)| \leq n$ iterations, the coloring computed by the layers of each $\mathcal{A}_c$ effectively perform n iterations of the 1-WL algorithm

when applied to any graph $G \in \mathcal{X}$. Due to the convergence behavior of the 1-WL algorithm, these additional iterations do not increase the expressiveness of the colorings computed by each $\mathcal{A}_c$, such that we can conclude for any graph $G \in \mathcal{X}$:

$$\forall c \in \mathbb{N}: \quad |\{v \in V(G) \mid f^{(n)}(v) = c\}| = |\{v \in V(G) \mid \text{1-WL}(G)(v) = c\}|,$$

which states that the colorings are equal for a bijection $\phi : \mathbb{N} \to \mathbb{N}$, such that we can infer that they are equally expressive for distinguishing non-isomorphism.

To prove that the collection $\mathcal{C}$ is 1-WL-Discriminating, we need to show two properties: 1) Each function in the collection is permutation invariant, and 2) For each pair of graphs in $\mathcal{X}$ distinguishable by the 1-WL isomorphism test, there must exist a function in the collection that also distinguishes the pair.

For the first property, by Definition 10 of GNNs, all functions computed by GNNs are permutation-invariant. Regarding the second property, consider $G_1, G_2 \in \mathcal{X}$ with $G_1 \not\simeq_{1\text{WL}} G_2$. Let C_1 and C_2 represent the final colorings computed by the 1-WL algorithm when applied to G_1 and G_2, respectively. Since $G_1 \not\simeq_{1\text{WL}} G_2$, there exists a color $c \in \mathbb{N}$ such that $\text{hist}_{G_1,C_1}(c) \neq \text{hist}_{G_2,C_2}(c)$. Since, we know that each $\mathcal{A}_c$ computes equally expressive colorings of G_1 and G_2, we know that there exists a $c' \in \mathbb{N}$, such that $\mathcal{A}_{c'}(G_1) \neq \mathcal{A}_{c'}(G_2)$. □

Similar to the proof in the previous subsection, we will use Lemma 21 to introduce the ability to construct GNNs that take in as input multiple GNNs and then apply a multilayer perceptron to the combined output. This insight is leveraged in the following two corollaries in the proof, as well as in the final proof.

Lemma 21 (GNN Composition)**.** Let $\mathcal{C}$ be a collection of functions computable by GNNs. Further, let $\mathcal{A}_1, \ldots, \mathcal{A}_n \in \mathcal{C}$ and $\text{MLP}^\bullet$ a suitable multilayer perceptron, then the function $\hat{\mathcal{A}}(\cdot) \coloneqq \text{MLP}(\mathcal{A}_1(\cdot), \ldots, \mathcal{A}_n(\cdot))$ is also computable by a GNN.

Proof of Lemma 21. Before we begin the proof, we briefly introduce two notations. For any $x \in \mathbb{R}^d$, we will use the notation $x[i]$ to indicate the i.th element of the vector x. Additionally, we indicate the merge and aggregation function used in layer t by $\mathcal{A}_i$ as $f^{(t)}_{\text{merge},i}$ and $f^{(t)}_{\text{agg},i}$. Similarly, we denote the Readout function as Readout_i and the input function of $\mathcal{A}_i$ as $f^{(0)}_i$.

We will prove the lemma by giving a construction of a GNN model computing $\hat{\mathcal{A}}$. For the ease of readability and to reduce the complexity of the subsequent construction, we assume that for all $\mathcal{A}_i$ its functions $f^{(t)}_{\text{merge},i}, f^{(t)}_{\text{agg},i}$ and Readout_i map into the one-dimensional space $\mathbb{R}$ for all layers t. With this assumption, we avoid the need for a formal notation of the number of dimensions each of these functions map to.

Let T be the maximum number of layers of all $\mathcal{A}_1, \ldots, \mathcal{A}_n$. We construct the GNN $\hat{\mathcal{A}}$ with T layers, with the input layer working as follows on an input graph G:

$$\forall v \in V(G): \ \hat{f}^{(0)}(v) \coloneqq \begin{bmatrix} f^{(0)}_1(v) \\ \vdots \\ f^{(0)}_n(v) \end{bmatrix},$$

and each other layer $0 < t \leq T$ utilizing the merge $\hat{f}^{(t)}_{\text{merge}}$ and aggregation $\hat{f}^{(t)}_{\text{agg}}$ functions as

constructed in the following:

$$\hat{f}^{(t)}_{\text{merge}}(\hat{f}^{(t-1)}(v),\ Agg) := \begin{bmatrix} f^{(t)}_{\text{merge},1}(\hat{f}^{(t-1)}(v)[1],\ Agg[1]) \\ \vdots \\ f^{(t)}_{\text{merge},n}(\hat{f}^{(t-1)}(v)[n],\ Agg[n]) \end{bmatrix}, \quad \text{and}$$

$$\hat{f}^{(t)}_{\text{agg}}(\{\!\{\hat{f}^{(t-1)}(w) \mid w \in \mathcal{N}(v)\}\!\}) := \begin{bmatrix} f^{(t)}_{\text{agg},1}(\{\!\{\hat{f}^{(t-1)}(w)[1] \mid w \in \mathcal{N}(v)\}\!\}) \\ \vdots \\ f^{(t)}_{\text{agg},n}(\{\!\{\hat{f}^{(t-1)}(w)[n] \mid w \in \mathcal{N}(v)\}\!\}) \end{bmatrix}.$$

Note that, not all $\mathcal{A}_i$ will be comprised of T layers, such that for these cases the functions $f^{(t)}_{\text{merge},i}$ and $f^{(t)}_{\text{agg},i}$ will not be defined for all $t \in [T]$. In these cases, we define the functions as follows:

$$f^{(t)}_{\text{merge},i}(\hat{f}^{(t-1)}(v),\ Agg) \coloneqq \hat{f}^{(t-1)}(v), \quad \text{and}$$
$$f^{(t)}_{\text{agg},i}(\{\!\{\hat{f}^{(t-1)}(w) \mid w \in \mathcal{N}(v)\}\!\}) \coloneqq 0.$$

This definition of $f^{(t)}_{\text{merge},i}$ and $f^{(t)}_{\text{agg},i}$ results in the fact that the representation computed in the last layer of $\mathcal{A}_i$ is forwarded to the last layer T of $\hat{\mathcal{A}}$. Finally, we construct the $\mathsf{Readout}$ function of $\hat{\mathcal{A}}$ as follows:

$$\mathsf{Readout}(\{\!\{\hat{f}^{(T)}(v) \mid v \in V(G)\}\!\}) \coloneqq \mathsf{MLP}^{\bullet} \circ \begin{bmatrix} \mathsf{Readout}_1(\{\!\{\hat{f}^{(T)}(v)[1] \mid v \in V(G)\}\!\}) \\ \vdots \\ \mathsf{Readout}_n(\{\!\{\hat{f}^{(T)}(v)[n] \mid v \in V(G)\}\!\}) \end{bmatrix}.$$

With this, the proof concludes. Note that this proof can easily be extended to work without the assumption of each function mapping into a one-dimensional space. □

As a consequence of the previous two lemmas, we find ourselves in a similar position as at the beginning of the proof in Section 4.1. Specifically, we have established, through Lemma 20, the existence of a collection C of functions that can be computed by GNNs and can effectively distinguish any pair of graphs that are also distinguishable by the $\mathsf{1\text{-}WL}$ algorithm. Furthermore, with Lemma 21, we have demonstrated that the composition of multiple GNNs and a multilayer perceptron remains computable by a single GNN. Consequently, we can use the same proofs of Lemmas 18 and 19 to derive the Corollaries 22 and 23.

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

Corollary 23. Let $\mathcal{C}$ be a collection of functions from $\mathcal{X}$ to $\mathbb{R}$ computable by GNNs such 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 GNNs such that for all $G \in \mathcal{X}$: $\varphi_{G^*}(G) = \mathbb{1}_{G \simeq_{\text{1WL}} G^*}$.

In conclusion, the corollaries establish the computability of the indicator function $\mathbb{1}_{G_1 \simeq_{\text{1WL}} G_2}$ over the set $\mathcal{X}$ by a GNN. Building upon these results, we can now utilize all our lemmas to prove the theorem, which states that any function $\mathcal{B}$ computable by a $\mathsf{1\text{-}WL{+}NN}$ can also be computed by a GNN.

Proof of Theorem 12. Let $\mathcal{B}$ be a function that works over $\mathcal{X}$ to $\mathbb{R}$ computed by a 1-WL+NN model. We will prove that $\mathcal{B}$ is GNN computable by decomposing the function and then argue that the decomposition is computable by a GNN model. For this let $G \in \mathcal{X}$ be an arbitrary input graph, we can decompose $\mathcal{B}(G)$ as follows:

$$\begin{aligned}
\mathcal{B}(G) &= \Big(\frac{1}{|\mathcal{X}/\!\simeq_{1\mathrm{WL}}(G)|} \sum_{G^* \in \mathcal{X}} \mathbb{1}_{G \simeq_{1\mathrm{WL}} G^*} \Big) \cdot \mathcal{B}(G) \\
&= \sum_{G^* \in \mathcal{X}} \frac{1}{|\mathcal{X}/\!\simeq_{1\mathrm{WL}}(G)|} \cdot \mathcal{B}(G) \cdot \mathbb{1}_{G \simeq_{1\mathrm{WL}} G^*} \\
&= \sum_{G^* \in \mathcal{X}} \frac{1}{|\mathcal{X}/\!\simeq_{1\mathrm{WL}}(G^*)|} \cdot \mathcal{B}(G^*) \cdot \mathbb{1}_{G \simeq_{1\mathrm{WL}} G^*} \qquad (4.8) \\
&= \sum_{G^* \in \mathcal{X}} \frac{\mathcal{B}(G^*)}{|\mathcal{X}/\!\simeq_{1\mathrm{WL}}(G^*)|} \cdot \varphi_{G^*}(G) \qquad (4.9)
\end{aligned}$$

where $\mathcal{X}/\!\simeq_{1\mathrm{WL}}(G)$ denotes the set of all graphs that are equivalent to G according to the $\simeq_{1\mathrm{WL}}$ relation. Since the decomposition is very similar to the one present in the proof of Theorem 11, we will only provide reasoning for the correctness of Equations (4.8) and (4.9). For all other equations, refer to the explanation provided in the proof of Theorem 11

Equation (4.8): Due to the output of the indicator function $\mathbb{1}$ being either 0 or 1, we can infer that the inner product of each summand can only be nonzero if G^* is indistinguishable by the 1-WL isomorphism test from G. Using Lemma 14, we know that for every graph $G^* \in \mathcal{X}$: if $G^* \simeq_{1\mathrm{WL}} G$, then $\mathcal{B}(G^*) = \mathcal{B}(G)$.

Equation (4.9): Utilizing Corollary 23, we can replace the indicator function with $\varphi_{G^*}(G)$.

In conclusion, we have decomposed the GNN function $\mathcal{B}(G)$ such that the only factors that depend on the input graph G are the functions φ_{G^*}, which take G as input. This observation implies that all other factors are constants. Consequently, we can reason that the entire decomposition can be computed by a multilayer perceptron with a single layer, which takes the output of all φ_{G^*} for all $G^* \in \mathcal{X}$, applied to the input graph G. The multilayer perceptron then multiplies each of these values with the constant $\frac{\mathcal{B}(G^*)}{|\mathcal{X}/\!\simeq_{1\mathrm{WL}}(G^*)|}$ and takes the sum. The existence of such a multilayer perceptron is evident, and when combined with Lemma 17, we can assert that this composition is 1-WL+NN computable.

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. □