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 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 Definition

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

Definition 2. Let $\mathcal{C}$ be a collection of permutation invariant functions from $\mathcal{X}^{n\times n}$ 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 ($G_1 \not\simeq_{1\text{WL}} G_2$), there exists a function $h \in \mathcal{C}$ such that $f(G_1) \neq f(G_2)$.

Definition 3. Let $\mathcal{C}$ be a collection of permutation invariant functions from $\mathcal{X}^{n\times n}$ 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 $\|\mathcal{A} - h_{\mathcal{A},\epsilon}\|_\infty := \sup_{G\in\mathcal{X}} |f(G) - h_{\mathcal{A},\epsilon}(G)| < \epsilon$

2 Theorems

In this thesis we concentrate on finite graphs. We therefore, let $\mathcal{X}$ be a XYS.

Theorem 4 (1-WL+NN $\subseteq$ GNN)**.** Let $\mathcal{C}$ be a collection of functions from $\mathcal{X}$ to $\mathbb{R}$ computed by 1-WL+NN and $\mathcal{X}$ finite set of graphs. If $\mathcal{C}$ is 1-WL-Discriminating, then there exists an augmentation $\mathcal{C}'$ of $\mathcal{C}$ that is computable by 1-WL+NN, such that for every function $\mathcal{A}$ computed by a GNN, $\mathcal{A} \in \mathcal{C}'$.

With this theorem we can conclude that every function computable by a GNN is also computable by a 1-WL+NN.

3 Proofs

Throughout this thesis we will concentrate on finite graphs, such that we let $\mathcal{X}$ be $\mathcal{X} := \{1, \dots, k\}$ for an arbitrary $k \in \mathbb{N}$. We will prove Theorem 4 by introducing a couple of small lemmas, which combined prove the theorem. In detail, in Lemma 5 we show the existence of collections computed by 1-WL+NN that are 1-WL-Discriminating. In Lemmas 6 to 8 we derive properties of 1-WL+NN functions we will use throughout Lemmas 9 to 11 with which we prove the theorem. We took great inspiration for Lemmas 9 to 11 from the proof presented in section 3.1 in the work of **?**.

Lemma 5. There exists a collection $\mathcal{C}$ of functions from $\mathcal{X}$ to $\mathbb{R}$ computed by 1-WL+NN that is 1-WL-Discriminating.

Proof. We consider the collection $\mathfrak{B}_k$ of functions computed by 1-WL+NN, where every $\mathcal{B} \in \mathfrak{B}_k$ is of the form $\mathcal{B}(\cdot) = \text{MLP} \circ f_{\text{enc}}(\cdot)$. Here MLP is an arbitrary multilayer perceptron mapping vectors from $\mathbb{N}^K$ to $\mathbb{R}$ and f the *counting-encoding* function. Further, let $G_1, G_2 \in \mathcal{X}^{n\times n}$ such that the 1-WL isomorphism test concludes non-isomorphic ($G_1 \not\simeq G_2$). We denote with $(C_\infty)_G$ the final coloring computed by the 1-WL algorithm when applied on G. Due to the 1-WL isomorphism test concluding $G_1 \not\simeq G_2$, there exists a color $c \in \mathbb{N}$ such that $(C_\infty)_{G_1}(c) \neq (C_\infty)_{G_2}(c)$. If we now consider as MLP the following function $\text{MLP} : \mathbb{N}^K \to \mathbb{R}, v \mapsto W \cdot v$ with $W \in \mathbb{N}^{1\times K}$ such that $W_{1,c} := 1$ and $W_{1,i} := 0$ for all $i \in [K] \setminus \{c\}$. Then we can conclude that $\mathcal{B}(G_1) \neq \mathcal{B}(G_2)$. Since G_1, G_2 are arbitrary, we can conclude the proof. $\square$

Lemma 6 (1-WL+NN Permuation Invariance)**.** Let $\mathcal{C}$ be a collection of functions computed by 1-WL+NN, then every function $\mathcal{B} \in \mathcal{C}$ is permutation-invariant.

Proof. Let $\mathcal{C}$ be a collection of functions computed by 1-WL+NN. Let $\mathcal{B}$ be an arbitrary function in $\mathcal{C}$, then $\mathcal{B}$ is comprised as follows: $\mathcal{B}(\cdot) = \mathrm{MLP} \circ f_{\mathrm{enc}} \circ \text{1-WL}(\cdot)$. Since, the 1-WL coloring algorithm is permutation-invariant, the overall function is permutation invariant. $\square$

Lemma 7 (1-WL+NN Equivariance)**.** Let $\mathcal{C}$ be a collection of functions computed by 1-WL+NN, then for every function $\mathcal{B} \in \mathcal{C}$ and every pair of graphs $G_1, G_2 \in \mathcal{X}^{n \times n}$: if $G_1 \simeq_{\mathrm{1WL}} G_2$ than $\mathcal{B}(G_1) = \mathcal{B}(G_2)$.

Proof. Let $\mathcal{C}$ be a collection of functions computed by 1-WL+NN. Let $\mathcal{B}$ be an arbitrary function in $\mathcal{C}$, then $\mathcal{B}$ is comprised as follows: $\mathcal{B}(\cdot) = \mathrm{MLP} \circ f_{\mathrm{enc}} \circ \text{1-WL}(\cdot)$. Let $G_1, G_2 \in \mathcal{X}^{n \times n}$ be arbitrary graphs with $G_1 \simeq_{\mathrm{1WL}} G_2$, then by definition of the relation $\simeq_{\mathrm{1WL}}$ we know that $\text{1-WL}(G_1) = \text{1-WL}(G_2)$. With this the equivalence follows immediatly. $\square$

Lemma 8 (Composition Lemma)**.** Let $\mathcal{C}$ be a collection of functions computed by 1-WL+NN. Further, $h_1, \dots h_n \in \mathcal{C}$ and MLP a multilayer perceptron, than the function $\mathcal{A}$ composed of $\mathcal{A}(\cdot) := \mathrm{MLP}(h_1(\cdot), \dots, h_n(\cdot))$ is also computable by 1-WL+NN.

Proof. Assume the above and let $f_1, \dots, f_n$ be the encoding functions, as well as $\mathrm{MLP}_1, \dots, \mathrm{MLP}_n$ be the multilayer perceptrons used by $h_1, \dots h_n$ 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 $\mathrm{MLP}_1, \dots, \mathrm{MLP}_n$ simultaneously on their respective section of the encoding vector of f^*, and applies afterwards the new MLP on the concatenation of the output of all MLP_i. See Figure 1 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.

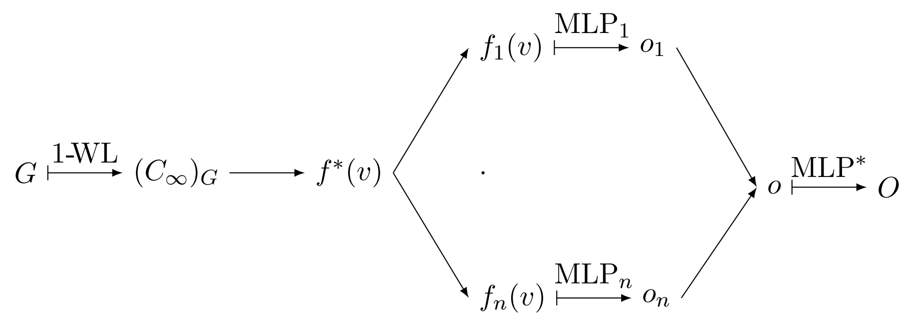

Figure 1: Sketch of the proof we use to prove lemma XYZ.

$\square$

Lemma 9. Let $\mathcal{C}$ be a collection of functions from $\mathcal{X}^{n \times n}$ to $\mathbb{R}$ computed by 1-WL+NN that is 1-WL-Discriminating. Then for all $G \in \mathcal{X}^{n \times n}$, there exists a function h_G computable by 1-WL+NN such that for all $G^* \in \mathcal{X}^{n \times n}$: $h_G(G^*) = 0$ if and only if $G \simeq_{\mathrm{1WL}} G^*$.

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

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

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

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

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

1. If $G_1 \simeq_{1\text{WL}} G^*$, then for every function $\overline{f}_{G_1,G_2}$ of the sum with $G_1 \not\simeq_{1\text{WL}} G_2$, we know, using Lemma 7, that $\overline{f}_{G_1,G_2}(G^*)$ is equal to $f_{G_1,G_2}(G_1)$ which is by definition 0, such that $h_{G_1}(G^*) = 0$.
2. If $G_1 \not\simeq_{1\text{WL}} G^*$, then $\overline{f}_{G_1,G^*}(G^*)$ is a summand of the overall sum, and since $\overline{f}_{G_1,G^*}(G^*) > 0$, than due to the non-negativity of each function $\overline{f}$ we can conclude $h_{G_1}(G*) > 0$.

This function can be encoded in an MLP by replacing the max terms of the last line in Equation 0.1 by the activation function ReLU. Therefore, we can conclude with Lemma 8 that for every graph G, h_G is also 1-WL+NN computable. □

Lemma 10. Let $\mathcal{C}$ be a collection of functions from $\mathcal{X}^{n\times n}$ to $\mathbb{R}$ computed by 1-WL+NN so that for all $G \in \mathcal{X}^{n\times n}$, there exists $h_G \in \mathcal{C}$ satisfying $h_G(G^*) = 0$ if and only if $G \simeq_{1\text{WL}} G^*$ for all $G^* \in \mathcal{X}^{n\times n}$. Then for every $G \in \mathcal{X}^{n\times n}$, there exists a function φ_G computable by 1-WL+NN such that for all $G^* \in \mathcal{X}^{n\times n}$: $\varphi_G(G^*) = \mathbb{1}_{G \simeq_{1\text{WL}} G^*}$.

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

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

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 2 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 $G \in \mathcal{X}^{n\times n}$ the function $\varphi_G(G^*) := \psi_{\delta_G}(h_G(G^*))$. 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 $\mathcal{X}^{n\times n}$, the input:

1. If $G \simeq_{1\text{WL}} G^*$, then $h_G(G^*) = 0$ resulting in $\varphi_G(G^*) = \psi_{\delta_G}(0) = 1$.
2. If $G \not\simeq_{1\text{WL}} G^*$ then $h_G(G^*) > 0$, such that $|h_G(G^*)| > \delta_G$ resulting in $\varphi_G(G^*) = 0$.

Note we can encode each φ_G via a single MLP layer, where δ_G is a constant and the max operator is replaced by the non-linear activation function ReLU of the layer. With Lemma 8 we can therefore conclude that φ_G is computable by 1-WL+NN for every graph $G \in \mathcal{X}^{n\times n}$. □

Lemma 11. Let $\mathcal{C}$ be a collection of functions from $\mathcal{X}^{n\times n}$ to $\mathbb{R}$ computed by 1-WL+NN so that for all $G \in \mathcal{X}^{n\times n}$, there exists $\varphi_G \in \mathcal{C}$ satisfying $\forall G^* \in \mathcal{X}^{n\times n} : \varphi_G(G^*) = \mathbb{1}_{G \simeq_{1\mathrm{WL}} G^*}$, then C is also GNN-Approximating.

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

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

with $\mathcal{X}^{n\times n}/\!\simeq_{1\mathrm{WL}}(G^*)$ denoting the set of all graphs G over $\mathcal{X}^{n\times n}$ 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 $G, H \in \mathcal{X}^{n\times n}$ with $G \simeq_{1\mathrm{WL}} H$: $\mathcal{A}(G) = \mathcal{A}(H)$. Therefore, we can decompose $\mathcal{A}$ as outlined above. We can encode this decomposition in a single MLP layer with $\frac{\mathcal{A}(G)}{|\mathcal{X}^{n\times n}/\simeq_{1\mathrm{WL}}(G)|}$ being a constant and $\varphi_G \in \mathcal{C}$ encoding the indicator function. Combined with the Lemma 8, 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 size of $\mathcal{X}^{n\times n}/\!\simeq_{1\mathrm{WL}}(G)$ well-defined for all graphs.

□

Appendix

Figures and graphs

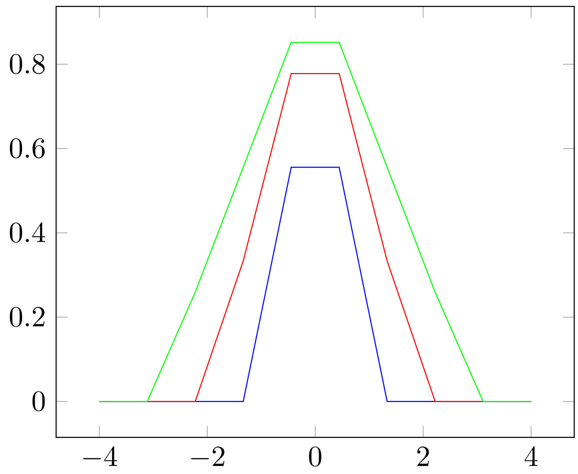

Figure 2: Illustration of the so-called "bump" function $\psi_a(x)$ with $a = 1$ in blue, $a = 2$ in red and $a = 3$ in green.

Proofs

Definition 12 (Multilayer Perceptron)**.** A multilayer perceptrons are a class of functions that map from R^n to R^m. We define it here as a sequence. Let MLP be a multilayer perceptron:

$$\begin{aligned}(\mathrm{MLP})_{i+1}(v) &:= \sigma(W_i \cdot (\mathrm{MLP})_i(v) + b_i)\\ (\mathrm{MLP})_1(v) &:= v\end{aligned}$$

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.

Lemma 8. Let $\mathcal{C}$ be a collection of functions computed by 1-WL+NN, $h_1, \ldots, h_n \in \mathcal{C}$, and $\mathrm{MLP}^\bullet$ 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 $G \in \mathcal{X}^{n \times n}$:

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

such that 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 conccat is the concatenation functions, concatenating all encoding vectors to one single vector.

Using the decomposition introduced in Definition 12, we can decompose each MLP_i at layer $j > 1$ 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:

$$\begin{aligned}
&(\mathrm{MLP}^*)_1(v) := v \\
&(\mathrm{MLP}^*)_{j+1}(v) := \sigma(W_j^* \cdot (\mathrm{MLP}^*)_j(v) + \mathsf{concact}(\begin{bmatrix} b_j^1 \\ \vdots \\ b_j^n \end{bmatrix})) && , \forall j \in [k] \\
&(\mathrm{MLP}^*)_{j+k+1}(v) := (\mathrm{MLP}^\bullet)_{j+1}(v) && , \forall j \in [k^\bullet - 1]
\end{aligned}$$

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 $\mathrm{MLP}^\bullet$ 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 $\mathrm{MLP}^\bullet$ 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. $\square$