Expressiveness of Line Graph Neural Networks

Jonas De Schouwer

Abstract

In this paper we characterize the expressivity of graph neural networks operating on line graphs in terms of their ability to distinguish non-isomorphic graphs. We introduce two new variations of the Weisfeiler-Leman test tailored to edge colourings, which we call the Line Weisfeiler-Leman Test (`lwl`) and the Weisfeiler-Leman Test with Edge Decoding (`wl-ed`). We prove that MPNNs operating on line graphs (LG-MPNNs) are at most as expressive as `lwl` and that there exists an LG-MPNN that is equally expressive as `lwl`, thereby establishing both an upper and a lower bound on their expressivity. We further show that `lwl` is strictly less expressive than `wl-ed`, which is in turn equally expressive as the standard Weisfeiler-Leman test. We also relate both variations to the well-studied k-WL hierarchy. Finally, we empirically compare the expressivities of MPNNs with an edge decoder and LG-MPNNs for the task of link prediction.

1 Introduction

Graphs are a natural way to structure data in many application domains, such as social networks, biological networks, recommendation systems, materials science and traffic networks. Following the many successes of deep learning, graph machine learning has emerged as a powerful tool to learn from graph-structured data. There are various tasks that are of interest, including classification, link prediction, property prediction, graph generation and community detection.

Recently, several works have investigated the use of line graphs in graph machine learning [1, 2, 3]. Line graphs offer a dual perspective, where nodes in the line graph correspond to edges in the original graph. This perspective is especially natural for the task of link prediction, as it converts a task on pairs of nodes to the simpler and better studied task of node classification.

The theoretical understanding of these models, however, is still in its infancy. Therefore, our objective in this study is to characterize the expressive power of graph neural networks operating on line graphs. Studying expressiveness can improve our understanding of the strengths and limitations of the model, thereby paving the way for improvements and guiding model selection for specific tasks. We focus specifically on message-passing neural networks operating on line graphs, henceforth called Line Graph MPNNs (LG-MPNNs). We theoretically characterize their expressiveness in terms of their ability to distinguish non-isomorphic graphs, and empirically measure it in terms of their efficacy for function approximation on a link prediction task.

2 Related Work

Graph Neural Networks Graph neural networks (GNNs) are a class of neural networks specifically designed to operate on graph-structured data [4]. One of the key challenges in designing GNNs is handling the inherent invariance of the graph domain w.r.t. permutations of the node ordering. The most commonly used class of GNNs are message passing neural networks (MPNNs) [5], which iteratively update node features by aggregating information from neighbouring nodes. Many GNN architectures fall under this framework, including the Basic GNN model [6], Graph Attention Networks [7], and Graph Isomorphism Networks [8].

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

Expressiveness The expressive power of a parametrized model refers to the range of functions and patterns that it can represent. In the context of graph neural networks, there are various ways to characterize expressiveness. The most common is the ability to distinguish non-isomorphic graphs, but function approximation [9], substructure counting [10], spectral decomposition [11] and logical representation [12] are other viable criterions. Our theoretical section concentrates on expressiveness in terms of graph distinguishability, while our empirical evaluation centers on function approximation, as this is easier to evaluate. It should be noted that [13] showed a theoretical equivalence between both notions of expressivity when the set of possible initial node labels is finite.

The usual way to quantify expressiveness in terms of graph distinguishability is through a comparison with various forms of the Weisfeiler-Leman (WL) test [14]. Importantly, MPNNs are known to be at most as expressive as WL [15], while [8] constructed a class of GNNs that is equally expressive as WL. Other works have introduced variations of the WL-test, such as the k-WL hierarchy [16] and the Relational Asymmetric Local 2-WL [17]. Our work aligns with this line of research by introducing two new variants of the WL-test tailored to edge colourings, by characterizing the expressive power of LG-MPNNs in terms of these variants, and by situating these variants in the k-WL hierarchy.

On the other hand, a number of works studied the expressiveness of GNNs in terms of their ability to approximate functions. [9] showed that GNNs can approximate any invariant function if one allows higher-order tensorization within the network, but that certain classes of functions require tensor orders as high as $\frac{n-2}{2}$ to be approximated. [18] later extended this result to the case of equivariant functions and GNNs. Our empirical evaluation also focuses on expressiveness in terms of function approximation, but we do not consider higher-order tensorization in our models.

Line Graph Neural Networks Most methods that use line graphs in graph machine learning do so in combination with the original graph [2, 19, 20, 21]. For instance, one approach [21] involves running GNNs concurrently on both the original graph and its line graph, and subsequently maximizing the mutual information between the two outputs. Further, various methods implicitely operate on the line graph by computing edge attributes, as exemplified by Generalized Message Passing [22]. Yet, a few approaches operate exclusively on the line graph, including LGLP [1] and INDIGO [3]. We limit the scope of this study to the latter family of approaches, as their analysis will also offer insights into the compound methodologies. To the best of our knowledge, there has been no theoretical exploration of these models to date.

3 Background

Graphs A *graph* is a tuple $G = (V, E, c)$ where V is a set of nodes, $E \subseteq V \times V$ is a set of edges, and $c : V \to \mathcal{C}$ is a node colouring. In this work, we consider undirected, simple graphs. We assume the set of possible node colours, $\mathcal{C}$, to be countable, so that we can leverage the theoretical results established in [8]. We denote the set of possible multisets of node colours by $\mathbb{N}^{\mathcal{C}}$. The reason for limiting ourselves to node-coloured graphs despite the focus on models that compute edge features is to fit in the existing literature on the Weisfeiler-Leman test and allow for a direct comparison. We write $u \in G$ as a shorthand for $u \in V(G)$. The *neighbourhood* of a node $u \in V$ is $\mathcal{N}(u) = \{v \in V \mid (u, v) \in E\}$. The *disjoint union* $G_1 \sqcup G_2$ of two graphs is the graph with node set $V(G_1) \sqcup V(G_2)$ and edge set $E(G_1) \sqcup E(G_2)$ that retains the colouring of the nodes. The *line graph* $L(G)$ of G is a graph whose nodes correspond to the edges of G, where we denote the node in $L(G)$ corresponding to the edge $(u, v) \in E(G)$ by $uv = vu$. Two nodes in $L(G)$ are connected if and only if the corresponding edges in G share a node. Finally, the node colouring of $L(G)$ is $c : V(L(G)) \to \mathcal{C} : uv \mapsto \xi(\{c(u), c(v)\})$, where ξ is an injective function from $\mathcal{C}^2$ to $\mathcal{C}$, i.e. two nodes in $L(G)$ get the same label iff the endpoints of their corresponding edges have the same set of labels in G. This is a variation of the atomic type defined in [15] tailored to undirected edges.

Message Passing Neural Networks A Message Passing Neural Network (MPNN) [5] is a framework for learning on graph-structured data that works by iteratively aggregating and updating node features. The feature of node u in iteration t is denoted by $\boldsymbol{x}_u^{(t)}$. It starts with $\boldsymbol{x}_u^{(0)} = c(u)$ and then iteratively updates the features of all nodes $u \in G$ as follows:

$$\boldsymbol{x}_u^{(t+1)} = \texttt{UPD}^{(t+1)}\left(\boldsymbol{x}_u^{(t)}, \texttt{AGG}^{(t+1)}\left(\boldsymbol{x}_u^{(t)}, \left\{\!\!\left\{\boldsymbol{x}_v^{(t)} \mid v \in \mathcal{N}(u)\right\}\!\!\right\}\right)\right) \tag{1}$$

Refinements and Graph Distinguishability A function $k(G) : V(G) \to \mathcal{C}$ *refines* a function $l(G) : V(G) \to \mathcal{C}$, denoted $k(G) \preceq l(G)$, if $k(G)(u) = k(G)(v)$ implies $l(G)(u) = l(G)(v)$ for all $u, v \in V(G)$. We call such functions *equivalent*, denoted $k(G) \equiv l(G)$, if they refine each other. We say that an algorithm $\mathcal{A}$ that generates node features on graphs *distinguishes* G_1 and G_2 if $\{\!\{\mathcal{A}(G_1)(u) \mid u \in V(G_1)\}\!\} \neq \{\!\{\mathcal{A}(G_2)(u) \mid u \in V(G_2)\}\!\}$. We write $\mathcal{A}(u)$ as a shorthand for $\mathcal{A}(G)(u)$ when it is clear which graph u belongs to.

The Weisfeiler-Leman Test The WL-test is a well-known graph isomorphism test that iteratively refines node colourings of graphs [14]. It starts by assigning each node $u \in G$ the colour $\texttt{wl}^{(0)}(G)(u) = c(u)$ and then iteratively updates the colour of each node u as follows:

$$\texttt{wl}^{(t+1)}(G)(u) = \tau\left(\texttt{wl}^{(t)}(G)(u), \left\{\!\!\left\{\texttt{wl}^{(t)}(G)(v) \mid v \in \mathcal{N}(u)\right\}\!\!\right\}\right) \tag{2}$$

where τ is an injective map from $\mathcal{C} \times \mathbb{N}^{\mathcal{C}}$ to $\mathcal{C}$.

4 Theory

In this section we study the expressive power of *Line Graph MPNNs* (LG-MPNNs), which we define as MPNNs (outlined in Section 3) applied to the line graph of an input graph. The Line GNNs proposed in [1] fall under this framework. We quantify their ability to distinguish non-isomorphic graphs by comparing them to variants of the WL-test. Because LG-MPNNs compute edge features of the original graph rather than node features, we devise two new variants of the standard WL-test that are specifically tailored to computing edge colourings.

We start by proving a lemma that establishes an elegant connection between refinements and graph distinguishability, which is often left implicit in the literature. Next, we introduce our two variants of the WL-test and compare their expressiveness to each other, to Line Graph MPNNs and to the standard WL-test. Finally, we situate both variants in the well-studied k-WL hierarchy.

4.1 A connection between refinements and graph distinguishability

We call an algorithm $\mathcal{A}$ that generates node features on graphs *union-preserving* if, for any graphs G_1, G_2 and node $u \in V(G_1 \sqcup G_2)$, the following holds:

$$\mathcal{A}(G_1 \sqcup G_2)(u) = \begin{cases} \mathcal{A}(G_1)(u) & \text{if } u \in V(G_1) \\ \mathcal{A}(G_2)(u) & \text{if } u \in V(G_2) \end{cases} \tag{3}$$

In particular, any algorithm that cannot transfer information across disconnected components is union-preserving. This includes WL, all MPNNs, all LG-MPNNs and the two WL-variants we will define in Subsection 4.2. On the other hand, algorithms that are not union-preserving include k-WL for $k > 1$ [16] and WL with global readout [12].

Lemma 1. *If $\mathcal{A}$ and $\mathcal{B}$ are union-preserving algorithms such that $\mathcal{A}(G) \preceq \mathcal{B}(G)$ for all graphs G, then $\mathcal{A}$ is at least as expressive as $\mathcal{B}$ for distinguishing non-isomorphic graphs.*

Proof. Suppose that $\mathcal{A}$ cannot distinguish between G_1 and G_2, i.e. $\{\!\{\mathcal{A}(G_1)(u) \mid u \in V(G_1)\}\!\} = \{\!\{\mathcal{A}(G_2)(v) \mid v \in V(G_2)\}\!\}$. Then we can choose node orderings $V(G_1) = \{u_1, \ldots, u_N\}$ and $V(G_2) = \{v_1, \ldots, v_N\}$ such that $\mathcal{A}(G_1)(u_i) = \mathcal{A}(G_2)(v_i)$ for all i. Because $\mathcal{A}$ and $\mathcal{B}$ are union-preserving and $\mathcal{A}(G_1 \sqcup G_2) \preceq \mathcal{B}(G_1 \sqcup G_2)$, it follows that $\mathcal{B}(G_1)(u_i) = \mathcal{B}(G_2)(v_i)$ as well for all i. Consequently, $\{\!\{\mathcal{B}(G_1)(u) \mid u \in V(G_1)\}\!\} = \{\!\{\mathcal{B}(G_2)(v) \mid v \in V(G_2)\}\!\}$, so $\mathcal{B}$ cannot distinguish between G_1 and G_2 either. □

4.2 Two variants of the Weisfeiler-Leman test for edge colourings

The first variant of the WL-test we introduce is the *Line Weisfeiler-Leman Test*, denoted `lwl`, which is a direct application of the standard WL-algorithm to the line graph. Concretely, we define the `lwl` algorithm as follows:

- Initialize the colour of each node $uv \in L(G)$ to $\texttt{lwl}^{(0)}(G)(uv) = \xi(\{c(u), c(v)\})$, where ξ is as outlined in Section 3.

- Iteratively update the colour of each node $uv \in L(G)$ as follows:

$$\texttt{lwl}^{(t+1)}(G)(uv) = \tau\left(\texttt{lwl}^{(t)}(G)(uv), \left\{\!\!\left\{\texttt{lwl}^{(t)}(G)(xy) \mid xy \in \mathcal{N}(uv)\right\}\!\!\right\}\right) \tag{4}$$

  where τ is an injective map from $\mathcal{C} \times \mathbb{N}^{\mathcal{C}}$ to $\mathcal{C}$. We remind the reader that $\mathcal{N}(uv)$ denotes the neighbourhood of uv in $L(G)$, i.e. $\mathcal{N}(uv) = \{ud_u \mid d_u \in \mathcal{N}_G(u)\} \cup \{vd_v \mid d_v \in \mathcal{N}_G(v)\}$.

The second variant we introduce is the *Weisfeiler-Leman Test with Edge Decoding*, denoted `wl-ed`, which enhances the standard WL-algorithm with an injective decoder $\chi : \mathcal{C}^2 \to \mathcal{C}$ to transform the node colours into edge colours. Concretely, it applies the WL-algorithm to the original node-coloured graph G, computing a node colouring $\texttt{wl}^{(t)}(G) : V(G) \to \mathcal{C}$ at each iteration t. Then the edge colouring at iteration t is computed as $\texttt{wl-ed}^{(t)}(G)(uv) = \chi(\{\texttt{wl}^{(t)}(G)(u), \texttt{wl}^{(t)}(G)(v)\})$.

In what follows, we first prove that LG-MPNNs are at most as expressive as `lwl` (Theorem 2) and that there exists an LG-MPNN that is equally expressive as `lwl` (Theorem 3). Afterwards, we show that `lwl` is less expressive than `wl-ed` (Theorem 4) and that `wl-ed` is equally expressive as the standard WL-test (Theorem 5). Theorems 4 and 5 are proven in the Appendices.

Theorem 2. *LG-MPNNs are at most as expressive as `lwl`.*

Proof. Previous work on the expressivity of MPNNs [15] has shown that, for any MPNN with T layers, and for all labelled graphs G and iterations $t \in \{0, \dots, T\}$, the WL-test's node colouring $\texttt{wl}^{(t)}(G)$ is a refinement of the MPNN's node features $\boldsymbol{x}^{(t)}(G)$: $\texttt{wl}^{(t)}(G) \preceq \boldsymbol{x}^{(t)}(G)$.

Applying this theorem to the line graph of G, one obtains that for any LG-MPNN with T layers, and for all labelled graphs G and iterations $t \in \{0, \dots, T\}$: $\texttt{lwl}^{(t)}(G) \preceq \boldsymbol{x}^{(t)}(G)$.

As both `lwl` and any LG-MPNN are union-preserving, it follows from Lemma 1 that LG-MPNNs are at most as expressive as the `lwl`-test. □

Theorem 3. *There exists an LG-MPNN that is equally expressive as `lwl`.*

Proof. Consider any finite set of graphs $\{G_1, \dots, G_N\}$ that are pairwise distinguishable by `lwl`. This is equivalent to saying that $\{L(G_1), \dots, L(G_N)\}$ are pairwise distinguishable by `wl`. [8] showed that this implies the existence of an MPNN that can distinguish the line graphs, thus the existence of an LG-MPNN that can distinguish the original set $\{G_1, \dots, G_N\}$. □

Theorem 4. *`lwl` is strictly less expressive than `wl-ed`.*

Theorem 5. *`wl-ed` is equally expressive as the standard WL-test.*

4.3 Connection to the k-WL hierarchy

A well-known variant of the standard WL-test is the (folklore) k-WL test for $k \in \mathbb{N}$, which iteratively refines colourings of all k-tuples of nodes [16]. This is often termed the k-WL hierarchy because the $(k+1)$-WL test is strictly more expressive than the k-WL test for every $k \geq 1$ [23]. The bottom of the hierarchy, the 1-WL test, is equivalent to the standard WL-test. Because we related LG-MPNNs, `lwl` and `wl-ed` to the standard WL-test, we also placed them in the k-WL hierarchy. In particular, LG-MPNNs and `lwl` are strictly less expressive than k-`wl` for every $k \geq 1$, while `wl-ed` is equally expressive as 1-`wl`, but strictly less expressive than k-`wl` for $k > 1$.

5 Experiments

In this section, we empirically compare the expressiveness of two architectures in terms of their ability to approximate functions for the task of link prediction. The studied architectures are an MPNN with an edge decoder (MPNN+ED) and an LG-MPNN. To compare their expressivities, we train instances of both architectures on link prediction tasks and compare their performances on the test set, as this is a measure of the model's ability to capture the underlying function in the data. Based on the theoretical results – i.e. Theorem 4, the existence of an MPNN equally expressive as the WL-test [8] and the theoretical equivalence between graph isomorphism testing and function approximation [13] – we hypothesize that MPNNs with an edge decoder are more expressive than LG-MPNNs, which will result in a higher performance on the test set.

Datasets We searched for datasets that satisfy the following criteria: (1) they consist of undirected, simple graphs with node features, (2) the link prediction task makes sense in the context of the dataset, and (3) the dataset is publicly available. Considering these criteria, we chose the Twitch EN dataset [24] and the protein-protein interaction (PPI) dataset [25]. In the Twitch dataset, nodes correspond to Twitch users and edges to mutual friendship, while node features are the user's gaming preferences, location and streaming habits. In the PPI dataset, nodes represent proteins and edges are interactions between them, while node features are positional gene sets, motif gene sets and immunological signatures. To preprocess these two datasets for the link prediction task, we followed the setup from LGLP [1]. This consists of sampling pairs of nodes (50% positive and 50% negative links) and extracting the 1-hop enclosing subgraph. Unlike [1], however, we store both the subgraph and its line graph, as we use both in our experiments. The two target nodes are connected to allow message passing between them, and each node feature is enhanced with the distance to both target nodes. Each such graph is endowed with a single target: 1 for positive links and 0 for negative links, so that the task becomes supervised binary graph classification. For PPI we sampled 5058 train graphs and 528 test graphs, where the train and test set come from different tissues in the human body. For the Twitch dataset, we sampled 3000 train graphs and 300 test graphs from the same social network.

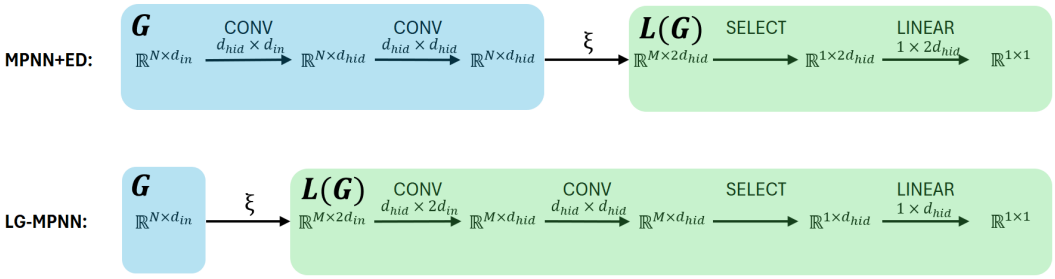


Figure 1: The architectures considered in this experiment

Architecture The two models we compare are instances of an MPNN with an edge decoder (MPNN+ED) and of an LG-MPNN, respectively. Their concrete architectures are visualized in Figure 1. Both models use a function ξ that transforms node features on G to node features on $L(G)$ by concatenating the features of both endpoints. This is not permutation invariant to the order of the endpoints, but we expect the models to learn this invariance from the training data. During inference, both models start from the input features on a graph G. The MPNN+ED model first performs two convolutions on G, then applies ξ to obtain edge features and finally selects the target link and applies a linear transformation to predict the link's existence. The edge decoder can be seen as the combination of ξ with the final linear transformation. On the other hand, the LG-MPNN model first applies ξ, then performs two convolutions on $L(G)$ and finally selects the target link and applies a linear layer to predict the link's existence. Both models were designed to be comparable. Yet, the LG-MPNN model has slightly more learnable parameters than the MPNN+ED model, because its first convolutional layer has twice the number of input features.

Hyperparameters and Metrics The hyperparameters for this experiment are summarized in Table 1. Most of these were taken from the implementation of LGLP [1], but we use only 80 epochs as experiments showed that this was sufficient for convergence. We examine three setups: the Twitch dataset with $d_{hid} = 20$ and the PPI dataset with $d_{hid} \in \{20, 52\}$. We evaluate the models in terms of test accuracy and AUC, which are standard metrics for binary classification tasks. The test performance quantifies how well the model was able to capture the underlying function in the dataset. Further, we report the training accuracy and AUC to give an indication of the capacity of the model. Finally, we measure the runtime of the models to give a relative indication of their efficiency.

Table 1: Hyperparameters

Hyperparameter	**Value**
Objective	BCE Loss
Optimizer	Adam
Learning Rate	$5 \cdot 10^{-3}$
Epochs	80
Batch Size	50
# Convolutions	2
Hidden Dimension	$20/52$
Nonlinearity	ReLU

Table 2 presents the results of the experiments. They reveal a marginally superior test performance for MPNN+ED compared to LG-MPNN, despite its slightly

Table 2: Experimental Results

Experiment	Model	Test Acc	Test AUC	Train Acc	Train AUC	Runtime
TwitchEN	MPNN+ED	91.3	93.8	100.0	99.8	23.1 min
$d_{hid} = 20$	LG-MPNN	91.0	81.9	99.9	88.3	105.1 min
PPI	MPNN+ED	79.7	85.2	93.2	98.5	42.1 min
$d_{hid} = 20$	LG-MPNN	75.8	80.0	96.6	99.3	85.7 min
PPI	MPNN+ED	75.4	77.4	100.0	99.3	37.6 min
$d_{hid} = 52$	LG-MPNN	75.0	77.5	99.6	98.4	96.3 min

smaller number of parameters. This supports our hypothesis that MPNN+ED is more expressive, although the difference is not very significant and further work is thus warranted to reach more conclusive results. Further, the training performance is near-perfect for both models, indicating that the models both have sufficient capacity to learn the underlying function. Unfortunately, this also means the difference is not significant enough to draw conclusions comparing the capacities of the models. Finally, the runtime of LG-MPNN is consistently higher than that of MPNN+ED, which is due to the considerably larger size of the line graph compared to the original graph. For instance, in PPI the average line graph has 1572 nodes and 59900 edges, compared to 92 nodes and 1572 edges for the average original graph. Based on the higher efficiency and slightly better performance of MPNN+ED, combined with the theoretical results from section 4 that guarantee a superior expressiveness compared to LG-MPNNs, we conclude that – at least in this setting with a simple, node-attributed graph – MPNN+ED is the better choice for the link prediction task.

6 Outlook and Conclusion

In this paper, we have introduced two new variations of the Weisfeiler-Leman test tailored to edge colourings: the Line Weisfeiler-Leman Test (`lwl`) and the Weisfeiler-Leman Test with Edge Decoding (`wl-ed`). We have theoretically characterized the power for graph distinguishability of LG-MPNNs, `lwl` and `wl-ed`, and placed them in the k-WL hierarchy. Finally, we have empirically compared an MPNN with an edge decoder to an LG-MPNN in terms of their ability to approximate functions for the task of link prediction. Our findings indicate that our hypothesis – that MPNNs with an edge decoder are more expressive than LG-MPNNs – is likely correct, but the results are not significant enough to draw definite conclusions.

To obtain more conclusive results, the experiment in our paper should be extended to harder and larger link prediction tasks, such that the influence of the model's expressiveness and capacity becomes more apparent from the test and training performance, respectively. Our suggestion is the PCQM-Contact benchmark [26], which is a link prediction task for quantum chemistry known to be difficult. Further, the experiment should be extended to a more diverse range of MPNN+ED and LG-MPNN instances to examine the impact of different design choices on the architectures' expressiveness and capacity. This can be done by varying the number of layers, the type of convolutional layer, the hidden dimension and the fuction ξ. In particular, various options for ξ should be considered, prioritizing those that are invariant to the order of the endpoints. Finally, the discrepancy between the number of learnable parameters in the two models should be eliminated to nullify its influence on the results. This can be done by adding a fixed preprocessing layer to the LG-MPNN model that halves the number of input features, by considering a different task with $d_{in} = 1$ or by constructing a function ξ for which the output dimension is the same as the input dimension.

The theoretical framework proposed in this paper can be a stepping stone for further research in several directions. First, `lwl` and `wl-ed` can be extended to edge-attributed graphs, directed graphs and knowledge graphs, and these extensions can be related to each other and the existing graph isomorphism tests in the literature. Second, the impact of different node features on the expressiveness of LG-MPNNs, MPNN+EDs, `lwl` and `wl-ed` can be studied, as has been done for normal MPNNs and the standard WL-test [27, 28]. Finally, our theoretical results can be further validated in practice by an alternative experiment that directly evaluates the power of the models in terms of graph distinguishability. For instance, [29] employs a synthetic dataset consisting of pairs of non-isomorphic graphs distinguishable by the WL-test. The same dataset can be used to empirically study the expressivity of LG-MPNNs and MPNN+EDs, in terms of graph distinguishability rather than function approximation.

A Proof of Theorem 4

The proof of this theorem consists of two parts. In Lemma 6 we show that `1wl` is less expressive than `wl-ed`, and in Lemma 7 we show that this difference is strict.

Lemma 6. *`1wl` is less expressive than `wl-ed`.*

Proof. First, we prove the following by induction on t. For any graph G and $t \in \mathbb{N}$:

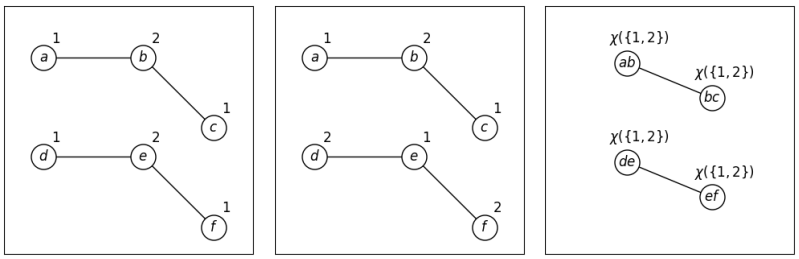
$$\texttt{wl-ed}^{(t)}(G) \preceq \texttt{1wl}^{(t)}(G) \tag{5}$$

Fix the graph G and shorten the notation $\texttt{wl}^{(t)} \coloneqq \texttt{wl}^{(t)}(G)$ for clarity (and analogous for `1wl` and `wl-ed`). The base case $t = 0$ is trivially true, because both $\texttt{wl-ed}^{(0)}$ and $\texttt{1wl}^{(0)}$ are obtained from the same initial node colourings with an injective edge decoder. For the induction step, assume (5) holds for t. Then take any $uv, xy \in L(G)$ for which $\texttt{wl-ed}^{(t+1)}(uv) = \texttt{wl-ed}^{(t+1)}(xy)$. Due to the injectivity of τ and the decoder χ, the following implications hold:

$$\begin{aligned}
&\texttt{wl-ed}^{(t+1)}(uv) = \texttt{wl-ed}^{(t+1)}(xy) \\
&\Rightarrow \left\{\texttt{wl}^{(t+1)}(u), \texttt{wl}^{(t+1)}(v)\right\} = \left\{\texttt{wl}^{(t+1)}(x), \texttt{wl}^{(t+1)}(y)\right\} \\
&\overset{\text{WLOG}}{\Rightarrow} \texttt{wl}^{(t+1)}(u) = \texttt{wl}^{(t+1)}(x) \wedge \texttt{wl}^{(t+1)}(v) = \texttt{wl}^{(t+1)}(y) \\
&\Rightarrow \begin{cases} \texttt{wl}^{(t)}(u) = \texttt{wl}^{(t)}(x) \wedge \left\{\!\left\{\texttt{wl}^{(t)}(d_u) \mid d_u \in \mathcal{N}(u)\right\}\!\right\} = \left\{\!\left\{\texttt{wl}^{(t)}(d_x) \mid d_x \in \mathcal{N}(x)\right\}\!\right\} \\ \texttt{wl}^{(t)}(v) = \texttt{wl}^{(t)}(y) \wedge \left\{\!\left\{\texttt{wl}^{(t)}(d_v) \mid d_v \in \mathcal{N}(v)\right\}\!\right\} = \left\{\!\left\{\texttt{wl}^{(t)}(d_y) \mid d_y \in \mathcal{N}(y)\right\}\!\right\} \end{cases} \\
&\Rightarrow \begin{cases} \texttt{wl-ed}^{(t)}(uv) = \texttt{wl-ed}^{(t)}(xy) \\ \left\{\!\left\{\texttt{wl-ed}^{(t)}(ud_u) \mid d_u \in \mathcal{N}(u)\right\}\!\right\} = \left\{\!\left\{\texttt{wl-ed}^{(t)}(xd_x) \mid d_x \in \mathcal{N}(x)\right\}\!\right\} \\ \left\{\!\left\{\texttt{wl-ed}^{(t)}(vd_v) \mid d_v \in \mathcal{N}(v)\right\}\!\right\} = \left\{\!\left\{\texttt{wl-ed}^{(t)}(yd_y) \mid d_y \in \mathcal{N}(y)\right\}\!\right\} \end{cases} \\
&\overset{(5)}{\Rightarrow} \begin{cases} \texttt{1wl}^{(t)}(uv) = \texttt{1wl}^{(t)}(xy) \\ \left\{\!\left\{\texttt{1wl}^{(t)}(ud_u) \mid d_u \in \mathcal{N}(u)\right\}\!\right\} = \left\{\!\left\{\texttt{1wl}^{(t)}(xd_x) \mid d_x \in \mathcal{N}(x)\right\}\!\right\} \\ \left\{\!\left\{\texttt{1wl}^{(t)}(vd_v) \mid d_v \in \mathcal{N}(v)\right\}\!\right\} = \left\{\!\left\{\texttt{1wl}^{(t)}(yd_y) \mid d_y \in \mathcal{N}(y)\right\}\!\right\} \end{cases} \\
&\Rightarrow \texttt{1wl}^{(t+1)}(uv) = \texttt{1wl}^{(t+1)}(xy)
\end{aligned}$$

Where the last step holds because $\mathcal{N}_{L(G)}(uv) = \{ud_u \mid d_u \in \mathcal{N}_G(u)\} \cup \{vd_v \mid d_v \in \mathcal{N}_G(v)\}$. We have thus shown that $\texttt{wl-ed}^{(t+1)}(G) \preceq \texttt{1wl}^{(t+1)}(G)$, which concludes the induction step. Because both `1wl` and `wl` are union-preserving, it follows from Lemma 1 that `1wl` is less expressive than `wl-ed`. □

Lemma 7. *There exist graphs G_1 and G_2 that are distinguishable by `wl-ed` but not by `1wl`.*



(a) G_1 (b) G_2 (c) $L(G_1) = L(G_2)$

Figure 2: Example of graphs G_1, G_2 that can be distinguished by `wl-ed` but not by `1wl`.

Proof. Consider the graphs G_1 and G_2 depicted in Figure 2. `wl-ed` can distinguish these graphs. To see this, note that the colour $\texttt{wl}^{(1)}(G_2)(e)$ does not occur anywhere in $\texttt{wl}^{(1)}(G_1)$, so clearly $\texttt{wl-ed}^{(1)}(G_2)(de) \notin \left\{\texttt{wl-ed}^{(1)}(G_1)(uv) \mid uv \in L(G_1)\right\}$ and hence

$$\left\{\!\left\{\texttt{wl-ed}^{(1)}(G_1)(uv) \mid uv \in L(G_1)\right\}\!\right\} \neq \left\{\!\left\{\texttt{wl-ed}^{(1)}(G_2)(uv) \mid uv \in L(G_2)\right\}\!\right\} \tag{6}$$

In contast, `1wl` can never distinguish between G_1 and G_2 because their line graphs are identical, as depicted in Figure 2c. We conclude that `1wl` is strictly less expressive than `wl-ed`. □

B Proof of Theorem 5

First, we show that the multiset of node labels generated by wl uniquely determines the multiset of line graph node labels generated by wl-ed (Lemma 8) and vice versa (9). Afterwards, we show how this implies Theorem 5. Our proofs build on the injectivity of τ and χ. The proofs are not constructive, but can be made so if the inverses of τ and χ on their codomains are known.

Lemma 8. *$\{\!\{ \texttt{wl}^{(t+1)}(u) \mid u \in G \}\!\}$ uniquely determines $\{\!\{ \texttt{wl-ed}^{(t)}(uv) \mid uv \in L(G) \}\!\}$ without requiring knowledge of the graph G.*

Proof. Due to the injectivity of τ, $M_1 = \{\!\{ \texttt{wl}^{(t+1)}(u) \mid u \in G \}\!\}$ uniquely determines $M_2 = \{\!\{ \left(\texttt{wl}^{(t)}(u), \{\!\{ \texttt{wl}^{(t)}(v) \mid v \in \mathcal{N}(u) \}\!\}\right) \mid u \in G \}\!\}$.

By iterating over M_2 and collecting the sets $\left\{\texttt{wl}^{(t)}(u), \texttt{wl}^{(t)}(v)\right\}$ in a new multiset, one uniquely obtains $M_3 = \{\!\{ \left\{\texttt{wl}^{(t)}(u), \texttt{wl}^{(t)}(v)\right\} \mid u \in G, v \in \mathcal{N}(u) \}\!\}$. On the other hand, note that the previous iteration goes over every edge exactly twice. Undoubling the elements in M_3 yields $M_4 = \{\!\{ \left\{\texttt{wl}^{(t)}(u), \texttt{wl}^{(t)}(v)\right\} \mid (u, v) \in E(G) \}\!\}$, which in turn uniquely determines $M_5 = \{\!\{ \texttt{wl-ed}^{(t)}(uv) \mid uv \in L(G) \}\!\}$ by applying χ to all elements. □

In other words, Lemma 8 proves the existence of a mapping $\mathcal{S} : \mathbb{N}^{\mathcal{C}} \to \mathbb{N}^{\mathcal{C}}$ such that, for all graphs G and iterations $t \geq 0$:

$$\mathcal{S}\left(\left\{\!\!\left\{ \texttt{wl}^{(t+1)}(u) \mid u \in G \right\}\!\!\right\}\right) = \left\{\!\!\left\{ \texttt{wl-ed}^{(t)}(uv) \mid uv \in L(G) \right\}\!\!\right\} \tag{7}$$

Lemma 9. *If $t > 0$, $\{\!\{ \texttt{wl-ed}^{(t)}(G)(uv) \mid uv \in L(G) \}\!\}$ uniquely determines $\{\!\{ \texttt{wl}^{(t)}(u) \mid u \in G \}\!\}$ without requiring knowledge of the graph G.*

Proof. Due to the injectivity of χ, $M_1 = \{\!\{ \texttt{wl-ed}^{(t)}(uv) \mid uv \in L(G) \}\!\}$ uniquely determines $M_2 = \{\!\{ \left\{\texttt{wl}^{(t)}(u), \texttt{wl}^{(t)}(v)\right\} \mid (u, v) \in E(G) \}\!\}$.

When flattening this multiset – i.e. creating a multiset that contains both x and y for each $\{x, y\}$ in M_2 – we deterministically obtain the multiset M_3 that contains $\texttt{wl}^{(t)}(u)$ exactly $\deg(u)$ times for each $u \in G$.

By $t > 0$ and the injectivity of τ, $\texttt{wl}^{(t)}(u)$ uniquely determines $\left|\{\!\{ \texttt{wl}^{(t-1)}(v) \mid v \in \mathcal{N}(u) \}\!\}\right| = \deg(u)$ for each $u \in G$. So, if we divide the number of occurences of $\texttt{wl}^{(t)}(u)$ in M_3 by $\deg(u)$ for each u, we deterministically obtain $M_4 = \{\!\{ \texttt{wl}^{(t)}(u) \mid u \in G \}\!\}$. □

In other words, Lemma 9 proves the existence of a mapping $\mathcal{T} : \mathbb{N}^{\mathcal{C}} \to \mathbb{N}^{\mathcal{C}}$ such that, for all graphs G and iterations $t > 0$:

$$\mathcal{T}\left(\left\{\!\!\left\{ \texttt{wl-ed}^{(t)}(uv) \mid uv \in L(G) \right\}\!\!\right\}\right) = \left\{\!\!\left\{ \texttt{wl}^{(t)}(u) \mid u \in G \right\}\!\!\right\} \tag{8}$$

Theorem 5 (restated)**.** *wl-ed is equally expressive as the standard WL-test.*

Proof. Suppose that wl cannot distinguish between G_1 and G_2, which implies $\{\!\{ \texttt{wl}^{(t+1)}(u) \mid u \in G_1 \}\!\} = \{\!\{ \texttt{wl}^{(t+1)}(u) \mid u \in G_2 \}\!\}$ for each t. Then applying $\mathcal{S}$ to each side retains the inequality. So $\{\!\{ \texttt{wl-ed}^{(t)}(uv) \mid uv \in L(G_1) \}\!\} = \{\!\{ \texttt{wl-ed}^{(t)}(uv) \mid uv \in L(G_2) \}\!\}$ as well, and hence that wl-ed cannot distinguish between G_1 and G_2 either.

Similarly, suppose that wl-ed cannot distinguish between G_1 and G_2, which implies $\{\!\{ \texttt{wl-ed}^{(t)}(uv) \mid uv \in L(G_1) \}\!\} = \{\!\{ \texttt{wl-ed}^{(t)}(uv) \mid uv \in L(G_2) \}\!\}$ for each $t > 0$. Then applying $\mathcal{T}$ to each side retains the inequality. So $\{\!\{ \texttt{wl}^{(t)}(u) \mid u \in G_1 \}\!\} = \{\!\{ \texttt{wl}^{(t)}(u) \mid u \in G_2 \}\!\}$ as well for each $t > 0$, and hence that wl cannot distinguish between G_1 and G_2 either. □

References

[1] Lei Cai, Jundong Li, Jie Wang, and Shuiwang Ji. Line graph neural networks for link prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5103–5113, 2021.

[2] Kamal Choudhary and Brian DeCost. Atomistic line graph neural network for improved materials property predictions. *npj Computational Materials*, 7(1):185, 2021.

[3] Shuwen Liu, Bernardo Grau, Ian Horrocks, and Egor Kostylev. Indigo: Gnn-based inductive knowledge graph completion using pair-wise encoding. *Advances in Neural Information Processing Systems*, 34:2034–2045, 2021.

[4] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[5] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[6] William L Hamilton. *Graph representation learning*. Morgan & Claypool Publishers, 2020.

[7] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. Graph attention networks. *stat*, 1050(20):10–48550, 2017.

[8] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[9] Haggai Maron, Ethan Fetaya, Nimrod Segol, and Yaron Lipman. On the universality of invariant networks. In *International conference on machine learning*, pages 4363–4371. PMLR, 2019.

[10] Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can graph neural networks count substructures? *Advances in neural information processing systems*, 33:10383–10395, 2020.

[11] Muhammet Balcilar, Guillaume Renton, Pierre Héroux, Benoit Gaüzère, Sébastien Adam, and Paul Honeine. Analyzing the expressive power of graph neural networks in a spectral perspective. In *International Conference on Learning Representations*, 2020.

[12] Pablo Barceló, Egor V Kostylev, Mikael Monet, Jorge Pérez, Juan Reutter, and Juan-Pablo Silva. The logical expressiveness of graph neural networks. In *8th International Conference on Learning Representations (ICLR 2020)*, 2020.

[13] Zhengdao Chen, Soledad Villar, Lei Chen, and Joan Bruna. On the equivalence between graph isomorphism testing and function approximation with gnns. *Advances in neural information processing systems*, 32, 2019.

[14] Boris Weisfeiler and Andrei Leman. The reduction of a graph to canonical form and the algebra which appears therein. *nti, Series*, 2(9):12–16, 1968.

[15] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4602–4609, 2019.

[16] Neil Immerman and Eric Lander. *Describing graphs: A first-order approach to graph canonization*. Springer, 1990.

[17] Xingyue Huang, Miguel Romero, Ismail Ceylan, and Pablo Barceló. A theory of link prediction via relational weisfeiler-leman on knowledge graphs. *Advances in Neural Information Processing Systems*, 36, 2024.

[18] Nicolas Keriven and Gabriel Peyré. Universal invariant and equivariant graph neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[19] Zhengdao Chen, Xiang Li, and Joan Bruna. Supervised community detection with line graph neural networks. *arXiv preprint arXiv:1705.08415*, 2017.

[20] Xiaodong Jiang, Pengsheng Ji, and Sheng Li. Censnet: Convolution with edge-node switching in graph neural networks. In *IJCAI*, pages 2656–2662, 2019.

[21] Zehua Zhang, Shilin Sun, Guixiang Ma, and Caiming Zhong. Line graph contrastive learning for link prediction. *Pattern Recognition*, 140:109537, 2023.

[22] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[23] Jin-Yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992.

[24] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-Scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 05 2021.

[25] Marinka Zitnik and Jure Leskovec. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33(14):i190–i198, 2017.

[26] Vijay Prakash Dwivedi, Ladislav Rampášek, Mikhail Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. Long range graph benchmark. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[27] Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Random features strengthen graph neural networks. In *Proceedings of the 2021 SIAM international conference on data mining (SDM)*, pages 333–341. SIAM, 2021.

[28] Or Feldman, Amit Boyarski, Shai Feldman, Dani Kogan, Avi Mendelson, and Chaim Baskin. Weisfeiler and leman go infinite: Spectral and combinatorial pre-colorings. *arXiv preprint arXiv:2201.13410*, 2022.

[29] Filippo Maria Bianchi and Veronica Lachi. The expressive power of pooling in graph neural networks. *Advances in Neural Information Processing Systems*, 36, 2024.