Geometric Deep Learning
Graph Classification with Spatial Graph Neural Networks

Lucas Kania
lucas.kania@usi.ch

Note The project presented in this report consists of the reproduction of selected experiments from the original paper[1] and further extensions to the proposed framework. Furthermore, many of the proofs presented in the original paper were re-done and simplified. Moreover, in the present work we will always refer to spatial graph neural networks and not to their spectral counterpart.

1 Introduction and Background

Graph structured data arises in chemistry, e.g. molecule structure (see figure 1), biological and social networks. The networks can have nodes rich in features or not. Specifically, in this work, we will experiment on networks whose features are not significant. Effectively learning from these networks requires a proper representation of the underlying graphs. Particularly, when the whole networks must be classified as pertaining to a certain group.

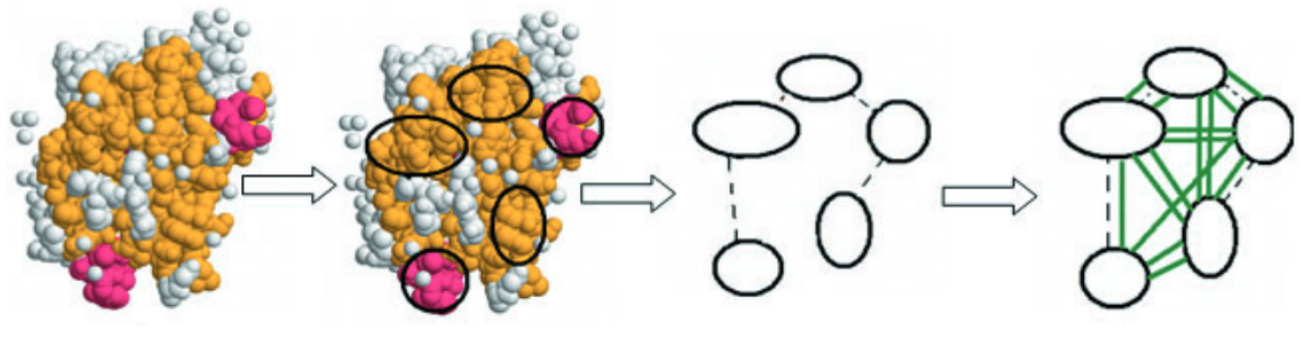


Figure 1: Graph derived from a protein's structure.

Recently, Graph Neural Network (GNN) have achieved state of the starts results in graph classification [2][3][4][5]. However, these developments have been mostly driven by intuition offering no guarantees, lower-bounds or upper-bounds with respect to the learning ability of the mentioned GNNs. Thus, they do not provide any understanding of the limits of a GNN in graph classification tasks.

Conversely, the present work develops a theoretical framework for distinguishing graphs (i.e. determining if two graphs are isomorphic or not), gives a precise upper-bound for the task (i.e.

the Weisfeiler-Lehman Graph Isomorphism Test (WL GI-TEST)) and states which kind of GNNs would be able to achieve the same performance.

2 Theoretical Framework

In this section, we first describe generally the inner workings of a GNN and connect it to the WL GI-TEST. Thereafter, we prove that the WL GI-TEST discrimination power is an upper-bound to the discrimination power of a GNN. Afterward, an optimal GNN architecture is derived, in the sense that it is as powerful as the WL GI-TEST for discriminating graphs.

2.1 Graph Neural Network

Given a graph $G = (V, E)$ and the input features of every node $\{X_v | v \in V\}$ a GNN works as follows.

The initial embedding of each node is determined by its input features

$$h_v^{(1)} = X_v \tag{1}$$

For each k-layer ($1 \leq k \leq K$), the neighborhood (i.e. a multiset[1] of the neighbors' embeddings $\{h_u^{(k-1)} | u \in N(v)\}$) of each node ($v$) is aggregated, resulting in an embedding for the neighborhood ($a_v^{(k)}$)

$$a_v^{(k)} = \text{AGGREGATE}^{(k)}(\{h_u^{(k-1)} | u \in N(v)\}) \tag{2}$$

Then, for each node, its embedding and the embedding of its neighborhood are combined, providing the new embedding

$$h_v^{(k)} = \text{COMBINE}^{(k)}(h_v^{(k-1)}, a_v^{(k)}) \tag{3}$$

This is enough for node classification tasks, which would use a linear projection after $h_v^{(K)}$ in order to predict the label of v. However, for graph classification tasks a final function (READOUT) over all the final embeddings of the nodes is needed in order to predict a label for the graph (h_G)

$$h_G = \text{READOUT}(\{h_v^{(K)} | v \in V\}) \tag{4}$$

Note that for the functions to be well-defined over multisets, they must be invariant to order since a multiset does not have any intrinsic order.

[1]A mutiset is a set where multiplicities of the same element are kept.

2.2 The Graph Isomorphism Problem

The Graph Isomorphism Problem (GIP) asks: Given two graphs G and G', does an isomorphism between them exist? (denoted $G \equiv G'$) That is, is it possible to relabel G's nodes in a way that we get G'?

2.2.1 Weisfeiler-Lehman Graph Isomorphism Test

The WL GI-TEST[2] is a heuristic[3] for solving the GIP. The test/algorithm proceeds as follows:

Initiate the labels of each node (l_v) with their degree

$$l_v = \text{degree}(v) \tag{5}$$

For each iteration, aggregate the neighborhood of each node and combine it with its label, to produce a new label for each node

$$a_v = \overline{\text{AGGREGATE}}(\{l_u | u \in N(v)\}) \tag{6}$$

$$l_v = \overline{\text{COMBINE}}(l_v, a_v) \tag{7}$$

where $\overline{\text{AGGREGATE}}$ and $\overline{\text{COMBINE}}$ are injective functions (i.e. $\forall a, b \in X, \quad a \neq b \Rightarrow f(a) \neq f(b)$). Thus, they preserve the distinctiveness of their inputs.

If the labels converge, that is, if from one iteration to the next one, no label changed. Then, the algorithm returns the multiset of the graph labels (WL(G) denotes the output of the algorithm for the graph G). Thereafter, it can be stated that

$$\text{WL}(G) \neq \text{WL}(G') \implies G \not\equiv G' \tag{8}$$

That is, if the multisets returned by the WL GI-TEST are different, there is no isomorphism between the graphs. The converse is not true.

Alternatively, instead of the degrees of the nodes, other input features (e.g. $l_v = X_v$) can be used.

[2]Here we present the one dimensional version of the WL GI-TEST [6].

[3]The WL GI-TEST is an heuristic since there are graphs for which the labels do not converge [6].

2.3 GNNs and the WL Test

From the descriptions of the previous sections, the similarity between a GNN and the WL GI-TEST is clear. In fact, any GNN is at most as powerful, with respect to its discriminative power, as the WL GI-TEST.

In order to prove this, first, we prove a small lemma that states that if two labels in the WL GI-TEST are equal, the corresponding embeddings in a GNN must be equal. For all the proofs in this report, G and G' have the same number of nodes and edges[4]. Moreover, $l_v^{(k)}$ denotes the label of the node v at the iteration k of the WL GI-TEST, while $h_v^{(k)}$ denotes the embedding of the node v at the kth-layer of the GNN.

Lemma 1. *Let $G = (V, E)$ and $G' = (V', E)$ be graphs, $GNN : \mathcal{G} \to \mathbb{R}^d$ a GNN.For any $v \in V$ and $v' \in V'$, $l_v^{(k)} = l_{v'}^{(k)} \implies h_v^{(k)} = h_{v'}^{(k)}$*

We will prove it by induction on the number (k) of iterations (for the WL GI-TEST) / layers (for the GNN).

Proof. if $k = 1$, then $\forall_{v \in V}\ l_v^{(1)} = X_v = h_v^{(1)}$ and $\forall_{v' \in V'}\ l_{v'}^{(1)} = X_{v'} = h_{v'}^{(1)}$. Thus, $l_v^{(1)} = l_{v'}^{(1)} \implies X_v = X_{v'} \implies h_v^{(1)} = h_{v'}^{(1)}$.

Given the inductive hypothesis (IH) $l_v^{(k)} = l_{v'}^{(k)} \implies h_v^{(k)} = h_{v'}^{(k)}$, we want to prove the test hypothesis $l_v^{(k+1)} = l_{v'}^{(k+1)} \implies h_v^{(k+1)} = h_{v'}^{(k+1)}$

If $l_v^{(k+1)} = l_{v'}^{(k+1)}$ since $\overline{\text{AGGREGATE}}$ is an injective function: $(l_v^{(k)}, \{l_u | u \in N(v)\}) = (l_{v'}^{(k)}, \{l_{u'} | u' \in N(v')\})$. Which implies, $l_v^{(k)} = l_{v'}^{(k)}$ and $\{l_u^{(k)} | u \in N(v)\} = \{l_{u'}^{(k)} | u' \in N(v')\}$.

We can directly apply the IH to the first equality, and recursively to the second multiset equality (since for every element in $\{l_u^{(k)} | u \in N(v)\}$, there must exist an element in $\{l_{u'}^{(k)} | u' \in N(v')\}$ s.t. they are equal).

Then, $(h_v^{(k)}, \{h_u | u \in N(v)\}) = (h_{v'}^{(k)}, \{h_{u'} | u' \in N(v')\}) \implies h_v^{(k)} = h_{v'}^{(k)}$ and $\{h_u | u \in N(v)\} = \{h_{u'} | u' \in N(v')\}$.

Thus, $\text{AGGREGATE}^{(k)}(\{h_u | u \in N(v)\}) = \text{AGGREGATE}^{(k)}(\{h_{u'} | u' \in N(v')\}) := A$. Hence,

$h_v^{(k+1)} = \text{COMBINE}^{(k)}(h_v^{(k)}, A) = \text{COMBINE}^{(k)}(h_{v'}^{(k)}, A) = h_{v'}^{(k+1)}$ □

From this point onwards $\text{WL}^{(k)}(G)$ denotes output of the WL GI-TEST at the iteration k, that is, a multiset of node labels. Additionally, $\text{GNN}^{(k)}(G)$ denote the output of a GNN at the kth-layer.

Theorem 1. *Let $G = (V, E)$ and $G' = (V', E')$ be graphs s.t. $G \not\equiv G'$, $GNN : \mathcal{G} \to \mathbb{R}^d$ a GNN. $WL^{(k)}(G) = WL^{(k)}(G') \implies GNN^{(k)}(G) = GNN^{(k)}(G')$*

Proof. $\text{WL}^{(k)}(G) = \text{WL}^{(k)}(G')$ implies that the multisets of labels are equal, i.e.
$\{(l_v^{(k)}, \{l_u^{(k)} | u \in N(v)\}) | v \in V\} = \{(l_{v'}^{(k)}, \{l_{u'}^{(k)} | u' \in N(v')\}) | v' \in V'\}$.

[4]If this wouldn't be the case, then it would be trivial to state that there is no isomorphism between the graphs.

Which implies that $\forall_{v \in V} \exists_{v' \in V'}$ s.t. $(l_v^{(k)}, \{l_u^{(k)} | u \in N(v)\}) = (l_{v'}^{(k)}, \{l_{u'}^{(k)} | u' \in N(v')\})$

Then $l_v^{(k)} = l_{v'}^{(k)}$ and $\{l_u^{(k)} | u \in N(v)\} = \{l_{u'}^{(k)} | u' \in N(v')\}$. Using first equality and the previous lemma, we have that $h_v^{(k)} = h_{v'}^{(k)}$. Furthermore, since the second equality can be reduced to the first one (as previously done), $(h_v^{(k)}, \{h_u^{(k)} | u \in N(v)\}) = (h_{v'}^{(k)}, \{h_{u'}^{(k)} | u' \in N(v')\})$.

Since this was done for an arbitrary v and v', we can state that $\forall_{v \in V} \exists_{v' \in V'}$ s.t. $(h_v^{(k)}, \{h_u^{(k)} | u \in N(v)\}) = (h_{v'}^{(k)}, \{h_{u'}^{(k)} | u' \in N(v')\})$. Which implies (since the graphs have the same number of nodes) that $\text{GNN}^{(k)}(G) = \text{GNN}^{(k)}(G')$ since READOUT is invariant to permutations. $\square$

We have proved that $\text{WL}^{(k)}(G) = \text{WL}^{(k)}(G') \implies \text{GNN}^{(k)}(G) = \text{GNN}^{(k)}(G')$, or equivalently, $\text{GNN}^{(k)}(G) \neq \text{GNN}^{(k)}(G') \implies \text{WL}^{(k)}(G) \neq \text{WL}^{(k)}(G')$. Thus, for any pair of graphs, if a GNN can differentiate them, then the WL GI-TEST can also do that. However, the converse is not true for all GNNs.

Now, we will prove under which conditions the converse is true. For this purpose, we will prove a small lemma first that will make the next proof easier.

Lemma 2. *Let $G = (V, E)$ and $G' = (V', E)$ be graphs, $GNN : \mathcal{G} \to \mathbb{R}^d$ a GNN with injective functions $AGGREGATE^{(k)}$ and $COMBINE^{(k)}$ $\forall_k$. For any $v \in V$ and $v' \in V'$, $h_v^{(k)} = h_{v'}^{(k)} \implies l_v^{(k)} = l_{v'}^{(k)}$*

The proof is by induction in the number (k) of iterations/layers.

Proof. If $k = 1$, $h_v^{(1)} = h_{v'}^{(1)} \implies X_v = X_{v'} \implies l_v^{(1)} = l_{v'}^{(1)}$.

The inductive hypothesis (IH) is $h_v^{(k)} = h_{v'}^{(k)} \implies l_v^{(k)} = l_{v'}^{(k)}$. The test hypothesis is $h_v^{(k+1)} = h_{v'}^{(k+1)} \implies l_v^{(k+1)} = l_{v'}^{(k+1)}$.

Given $h_v^{(k+1)} = h_{v'}^{(k+1)}$, we know that $(h_v^{(k)}, a_v^{(k)}) = (h_{v'}^{(k)}, a_{v'}^{(k)})$ since $\text{COMBINE}^{(k)}$ is a injective function. Thus, $h_v^{(k)} = h_{v'}^{(k)}$ and $a_v^{(k)} = a_{v'}^{(k)}$. Furthermore, since $\text{AGGREGATE}^{(k)}$ is injective, $\{h_u^{(k)} | u \in N(v)\} = \{h_{u'}^{(k)} | u' \in N(v')\}$.

Now using the IH we can state that, $l_v^{(k)} = l_{v'}^{(k)}$ and $\{l_u^{(k)} | u \in N(v)\} = \{l_{u'}^{(k)} | u' \in N(v')\}$. Which implies, $l_v^{(k+1)} = l_{v'}^{(k+1)}$. $\square$

Theorem 2. *Let $G = (V, E)$ and $G' = (V', E')$ be graphs s.t. $G \not\equiv G'$, $GNN : \mathcal{G} \to \mathbb{R}^d$ a GNN with injective functions $AGGREGATE^{(k)}$, $COMBINE^{(k)}$ and $READOUT^{(k)}$ $\forall_k$.*
$WL^{(k)}(G) \neq WL^{(k)}(G') \iff GNN^{(k)}(G) \neq GNN^{(k)}(G')$

Proof. The previous theorem has proven that $\text{WL}^{(k)}(G) \neq \text{WL}^{(k)}(G') \implies \text{GNN}^{(k)}(G) \neq \text{GNN}^{(k)}(G')$ for GNNs using any kind of function. Hence here we will only prove, $\text{GNN}^{(k)}(G) \neq \text{GNN}^{(k)}(G') \implies \text{WL}^{(k)}(G) \neq \text{WL}^{(k)}(G')$.

$\text{GNN}^{(k)}(G) \neq \text{GNN}^{(k)}(G')$ implies that $(h_v^{(k)}, \{h_u^{(k)} | u \in N(v)\}) \neq (h_{v'}^{(k)}, \{h_{u'}^{(k)} | u' \in N(v')\})$ since $\text{READOUT}^{(k)}$ is injective. Then, $\exists_{v \in V} \forall_{v' \in V'}$ s.t. $(h_v^{(k)}, \{h_u^{(k)} | u \in N(v)\}) \neq (h_{v'}^{(k)}, \{h_{u'}^{(k)} | u' \in N(v')\})$.

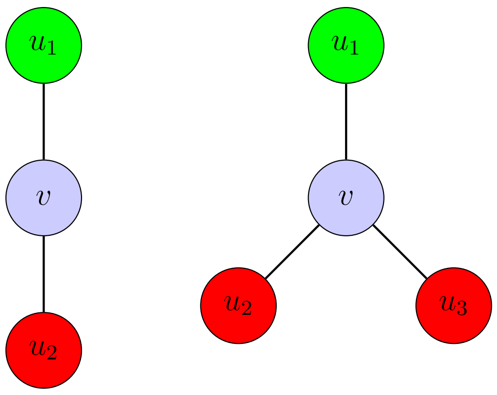
This could be because (1) $h_v^{(k)} \neq h_{v'}^{(k)}$ or (2) $\{h_u^{(k)} | u \in N(v)\} \neq \{h_{u'}^{(k)} | u' \in N(v')\}$. In any case, using the previous lemma we can state that $\exists_{v \in V} \forall_{v' \in V'}$ s.t. $(l_v^{(k)}, \{l_u^{(k)} | u \in N(v)\}) \neq (l_{v'}^{(k)}, \{l_{u'}^{(k)} | u' \in N(v')\})$. Thus, $\mathrm{WL}^{(k)}(G) \neq \mathrm{WL}^{(k)}(G')$. $\square$

2.4 Aggregation Functions

In the previous section, we proved that for a GNN to be as powerful as the WL GI-TEST, all its functions must be injective and well-defined over multisets. Here, we provide an intuition about that statement.

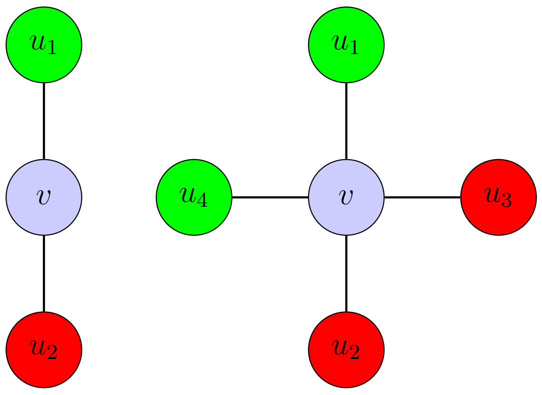
Given a node v, let's consider three cases for the AGGREGATE function (i.e. the merge of the neighbors' embeddings):(1) the maximum, (2) the mean and (3) the summation.

Assuming the input is one hot encoded, the maximum would not be able to distinguish these two graphs.



Hence, it is not an injective function over multisets. This is because the maximum only captures the presence of a feature and not the number of times it appeared.

Although, the mean would be able to distinguish the previous case, it wouldn't be able to do so in this one



since the mean only takes into account the proportions of the features.

Finally, the summation would always be able to distinguish these cases, since it takes into account the multiplicities of the neighbors' features. Thus, it is injective and well-defined over multisets.

2.5 Optimal GNN

In this section, we will find the required injective functions to build an optimal GNN. That is, one that can distinguish any pair of graphs that the WL GI-TEST can distinguish.

In order to build well-defined injective functions over multisets, first, we need to be sure that the input space of every layer of the GNN is countable.[5]

Lemma 3. *Given a countable input space $\mathbb{X}$, and $\mathcal{X}$ the set of bounded-size multisets of elements from $\mathbb{X}$, a GNN: $\mathcal{X} \to Y$ with injective functions AGGREGATE$^{(k)}$, COMBINE$^{(k)}$ and READOUT$^{(k)}$ $\forall_k$. Then, $\forall_k$ the range of GNN$^{(k)}$ is countable.*

Proof. Given a particular k, since GNN$^{(k)}$ is an injective function, if its domain is countable, its range is countable.

Since $\mathbb{X}$ is countable, $\mathbb{X}^r$ is countable for $r \in \mathbb{N}$. Furthermore, if we introduce a dummy element e, $\mathbb{X} \cup \{e\}$ is countable. Hence, $(\mathbb{X} \cup \{e\})^r$ is countable.

We choose $r = \max_{x \in \mathcal{X}} |x|$, where $|\cdot|$ denotes the number of elements of the multiset. Then, we define the injective function $h : \mathcal{X} \to (\mathbb{X} \cup \{e\})^r$, $h(x) = (x_1, \cdots, x_n, e, \cdots, e)$ where $(r - n)$ e-elements have been added to the end.

Since h is an injective mapping and $(\mathbb{X} \cup \{e\})^r$ is countable, $\mathcal{X}$ is countable. Therefore, the range of GNN$^{(k)}$ is countable.

Since the analysis was done for an arbitrary k, $\forall_k$ the range of GNN$^{(k)}$ is countable. □

Knowing that the input space of any layer is countable, we can properly define injective functions over it.

Lemma 4. *Given a countable space $\mathbb{X}$, $\exists f : \mathbb{X} \to \mathbb{N}^r$ s.t. $\exists$ injective $h : \mathcal{X} \to \mathbb{N}^n$ $h(x) := \sum_i^{|x|} f(x_i)$.*

Proof Sketch 1. *Since $\mathbb{X}$ is countable, then $\exists\ Z : \mathbb{X} \to \mathbb{N}$.*

We define $f : \mathbb{X} \to \mathbb{N}^r$, where $r = \max_{x \in \mathcal{X}} |x|$, as $f(x) := (0, \cdots, \underset{\text{in the } Z(x) \text{position}}{1}, \cdots, 0)$

Clearly f is injective. Furthermore, since f is a one-hot encoding, $h(x) := \sum_i^{|x|} f(x_i)$ is injective. □

Moreover, given the previous lemma, we know that any function $g : \mathcal{X} \to \mathbb{R}^n$ can be decomposed into $g = \phi \circ h$ for some function ϕ due to the injectivity of h.

Finally, we arrived at the desired corollary.

Corollary 1. *Given a countable input space $\mathbb{X}$, $\exists f : \mathbb{X} \to \mathbb{N}^r$ s.t. $\exists$ injective $p : (\mathbb{X}, \mathcal{X}) \to \mathbb{R}^n$ $p(x, N) := \epsilon f(x) + \sum_i^{|N|} f(N_i)$ where $\epsilon \in \mathbb{I}$.*

[5]This lemma closely follows the original proof.

Proof Sketch 2. *We use the f defined in the previous lemma. Given $(x, N) \neq (x', N')$, we want to show that $p(x, N) \neq p(x', N')$ (i.e. p is injective).*

If $(x, N) \neq (x', N')$, there are three cases: (1) $x = x$' and $N \neq N'$, (2) $x \neq x'$ and $N = N'$, and (3) $x \neq x'$ and $N \neq N'$.

For (1), $x = x' \implies f(x) = f(x')$ due to f injectivity. Moreover, by the previous lemma, we know that $\sum_i^{|N|} f(N_i)$ is injective, then $N \neq N' \implies \sum_i^{|N|} f(N_i) \neq \sum_i^{|N'|} f(N_i')$. Therefore, since adding an equality to an inequality doesn't change the inequality, $\epsilon f(x) + \sum_i^{|N|} f(N_i) \neq f(x') + \sum_i^{|N'|} f(N_i')$. Which implies, $p(x, N) \neq p(x', N')$. The second case is analogous.

For (3), assume $p(x, N) = p(x', N')$. Then $\epsilon f(x) + \sum_i^{|N|} f(N_i) = f(x') + \sum_i^{|N'|} f(N_i')$. Which can be rewritten as $\epsilon(f(x) - f(x')) = \sum_i^{|N'|} f(N_i') - \sum_i^{|N|} f(N_i)$. The right hand side of the equation belongs to $\mathbb{Z}^r$, while the left hand side belongs to $\mathbb{I}^r$. Thus, we have an absurd. Hence, $p(x, N) \neq p(x', N')$.

For all cases $p(x, N) \neq p(x', N')$. □

2.5.1 GNN Architecture

Given the previous corollary, we can now state that for any function $g : (\mathbb{X}, \mathcal{X}) \to \mathbb{R}^n$, we can decompose it into $g = \phi \circ p$ for some function ϕ due to the injectivity of p.

We can observe that $g(x, N) : (\phi \circ p)(x, N) = \phi(\epsilon f(x) + \sum_i^{|N|} f(N_i))$ can be easily converted to a GNN. Where, the composition of ϕ and p is approximated by a MLP.

$$a_v^{(k)} = \text{AGGREGATE}^{(k)}(\{h_u^{(k-1)} | u \in N(v)\}) = \sum_{u \in N(v)} h_u^{(k-1)} \tag{9}$$

$$h_v^{(k)} = \text{COMBINE}^{(k)}(h_v^{(k-1)}, a_v^{(k)}) = \text{MLP}^{(k)} \left(\epsilon h_v^{(k-1)} + a_v^{(k)}\right) \tag{10}$$

Since for graph classification we need a readout function,

$$h_G = \text{READOUT}(\{h_v^{(K)} | v \in V\}) = \sum_k^K W \sum_{v \in V} h_v^{(k)} \tag{11}$$

where we use the projected sum of the scores at every layer.

3 Experiments

3.1 Implementation

The implementation of the optimal GNN can be found in the file GIN.ipynb. The COMBINE and READOUT functions are implemented as described in section 2.5.1, while for the AGGREGATE function, the summation, maximum and mean aggregation functions (described in section 2.4) were implemented.

3.2 Training

The training accuracy was computed for the datasets MUTAG, PROTEINS, NCI1 and IMBD-BINARY[6]. The parameters used were: 5 GNN layers, 2-layer MLPs, 64 hidden dimension, mini-batch size of 128, 0.5 dropout ratio and 350 epochs of 50 iterations (i.e. a total of 17500 iterations). All experiments were cross-validated for each dataset using 10 folds.

[6]The details of the datasets can be found in [1].

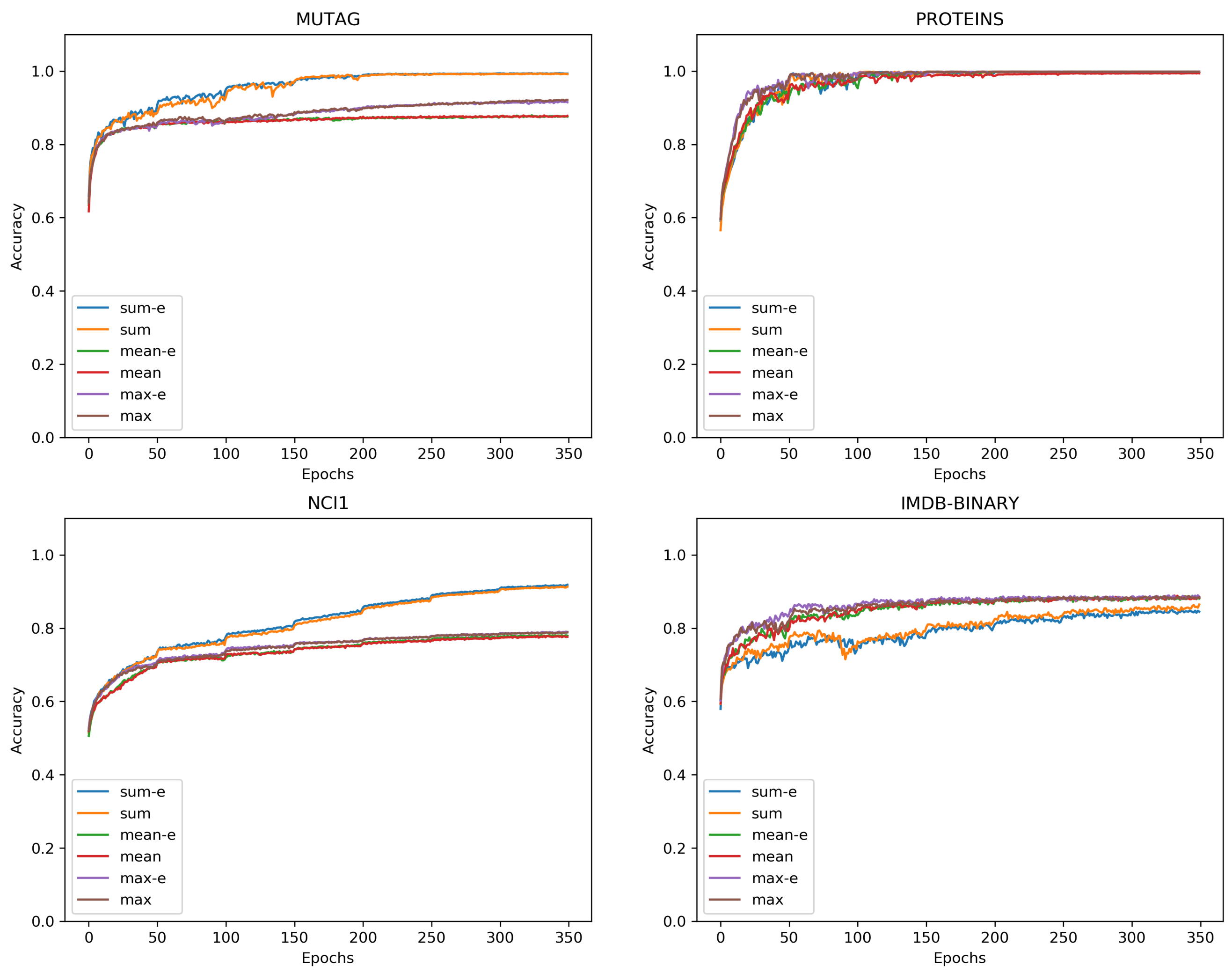


Figure 2: Average (over 10 runs) training accuracies for the different GNN architectures. The labels of the time-series indicate the AGGREGATE function used, and whether ϵ was learned or set to one.

Figure 2 and table 1 show that for the dataset PROTEINS, all methods perform approximately the same. However, for MUTAG and NCI1, the summation AGGREGATE function provided a clear advantage (the accuracy increased between 7 to 13 points). Conversely, for the dataset IMDB-BINARY all methods that didn't use the summation AGGREGATE function obtained better results (the accuracy was around two points lower than the other methods).

From these results, it can be stated that during training the AGGREGATE function performs (on average) equally or better than its non-injective counterparts (i.e mean and maximum) as expected (given that summation is more powerful than the others, see section 2.4).

Differences with the Original Work In the original work [1], only one run for each one of the datasets was shown, while in figure 2 we present the average training accuracy over 10 runs. On the PROTEINS dataset, only their implementation of the summation AGGREGATE function was able to overfit the dataset, here all aggregation functions can overfit it. Conversely, for the IMBD-BINARY dataset, their implementation using the summation aggregate is able to overfit the dataset while ours is not. However, they only presented one training run.

Name	MUTAG	PROTEINS	NCI1	IMDB-BINARY
sum-e	99.289 ± 0.419	99.848 ± 0.116	91.831 ± 0.757	84.500 ± 1.622
sum	99.164 ± 0.406	99.817 ± 0.130	91.295 ± 0.681	86.442 ± 2.207
mean-e	87.552 ± 1.244	99.826 ± 0.087	77.906 ± 1.135	88.050 ± 1.392
mean	87.773 ± 0.891	99.376 ± 0.221	77.635 ± 1.195	88.280 ± 2.141
max-e	91.485 ± 1.010	99.839 ± 0.115	78.945 ± 1.154	88.556 ± 1.810
max	92.124 ± 0.869	99.819 ± 0.123	78.830 ± 1.115	88.408 ± 1.413

Table 1: Average training accuracies over the last 50 iterations cross-validated across 10 folds.

3.3 Generalization

Table 3.3 displays the cross-validated test results (using 10 folds) for each one of the datasets. With the exception of the dataset PROTEINS, all obtained test accuracies are in correspondence with the results presented in [1] (taking into account the standard deviations).

Name	MUTAG	PROTEINS	NCI1	IMDB-BINARY
sum-e	85.228 ± 4.641	66.579 ± 4.093	79.658 ± 1.488	73.600 ± 1.625
sum	87.784 ± 6.256	63.241 ± 4.319	80.708 ± 1.772	71.900 ± 4.011
mean-e	82.374 ± 5.880	65.953 ± 3.269	76.107 ± 2.763	71.100 ± 4.230
mean	79.175 ± 6.321	64.332 ± 3.955	76.400 ± 1.515	71.200 ± 4.622
max-e	79.149 ± 9.861	66.035 ± 3.410	76.519 ± 2.035	72.700 ± 3.635
max	77.678 ± 3.827	64.228 ± 5.642	75.621 ± 1.472	71.400 ± 3.929

Table 2: Average test accuracies cross-validated across 10 folds.

We can observe that, as expected, for those datasets where the training performance of all GNN variations was similar (PROTEINS and IMDB-BINARY), the test accuracies were similar. For the other datasets (MUTAG and NCI1), the summation AGGREGATE function performed better at test time, as it did during training time.

4 Extensions

In section 2.5.1, due to corollary 1, an irrational number (i.e. ϵ) must be used in the COMBINE function. Instead of learning it (e.g. sum-e)[7], an approximation could be used. For instance, $\epsilon = \sqrt{2}$. Figure 3, shows the resulting training runs. Additionally, the test accuracies are presented in table 4.

[7]As proposed by the original authors [1].

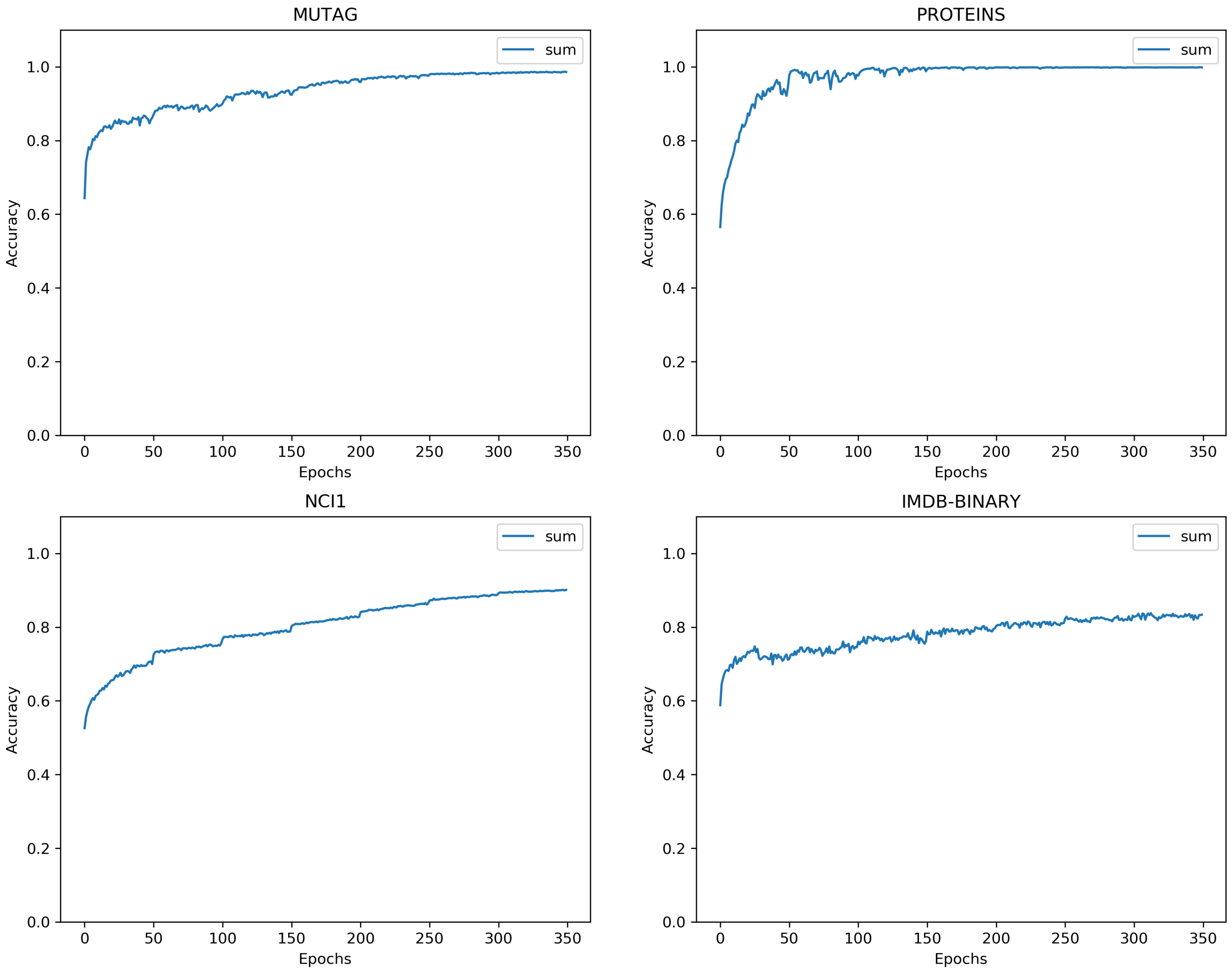


Figure 3: Average training accuracies (over 10 runs) of the GNN architecture described in section 2.5.1 but using $\sqrt{2}$ instead of ϵ.

Phase	MUTAG	PROTEINS	NCI1	IMDB-BINARY
Training	98.580±0.482	99.815±0.129	90.122±0.841	83.369±1.906
Test	86.284±6.099	66.397±2.512	80.534±1.638	73.900±3.673

Table 3: Average training and test accuracies (croos-validated over 10 runs) of the GNN architecture described in section 2.5.1 but using $\sqrt{2}$ instead of ϵ.

From the results we can see that sum-e, sum and sum-$\sqrt{2}$ perform comparatively the same. Thus, the use of an irrational number doesn't seem to be an important matter for learning.

Based on these results, since the irrational number is used only because of the proof of corollary 1, a different injective function could be used. In particular, instead of using $p(x, N) := \epsilon f(x) + \sum_i^{|N|} f(N_i)$ in the corollary 1, we could use $\psi(x, N) := 2^{f(x)} 3^{\sum_i^{|N|} f(N_i)}$ which is also injective due to the unique prime factorization (i.e. coordinate-wise unique prime factorization in this case). Furthermore, $\log \psi(x, N) = \log(2) f(x) + \log(3) \sum_i^{|N|} f(N_i)$ is an injective function. Thus, an analogous proof to corollary 1 can be done for it.

Figure 4 and table 4 present the training and testing results.

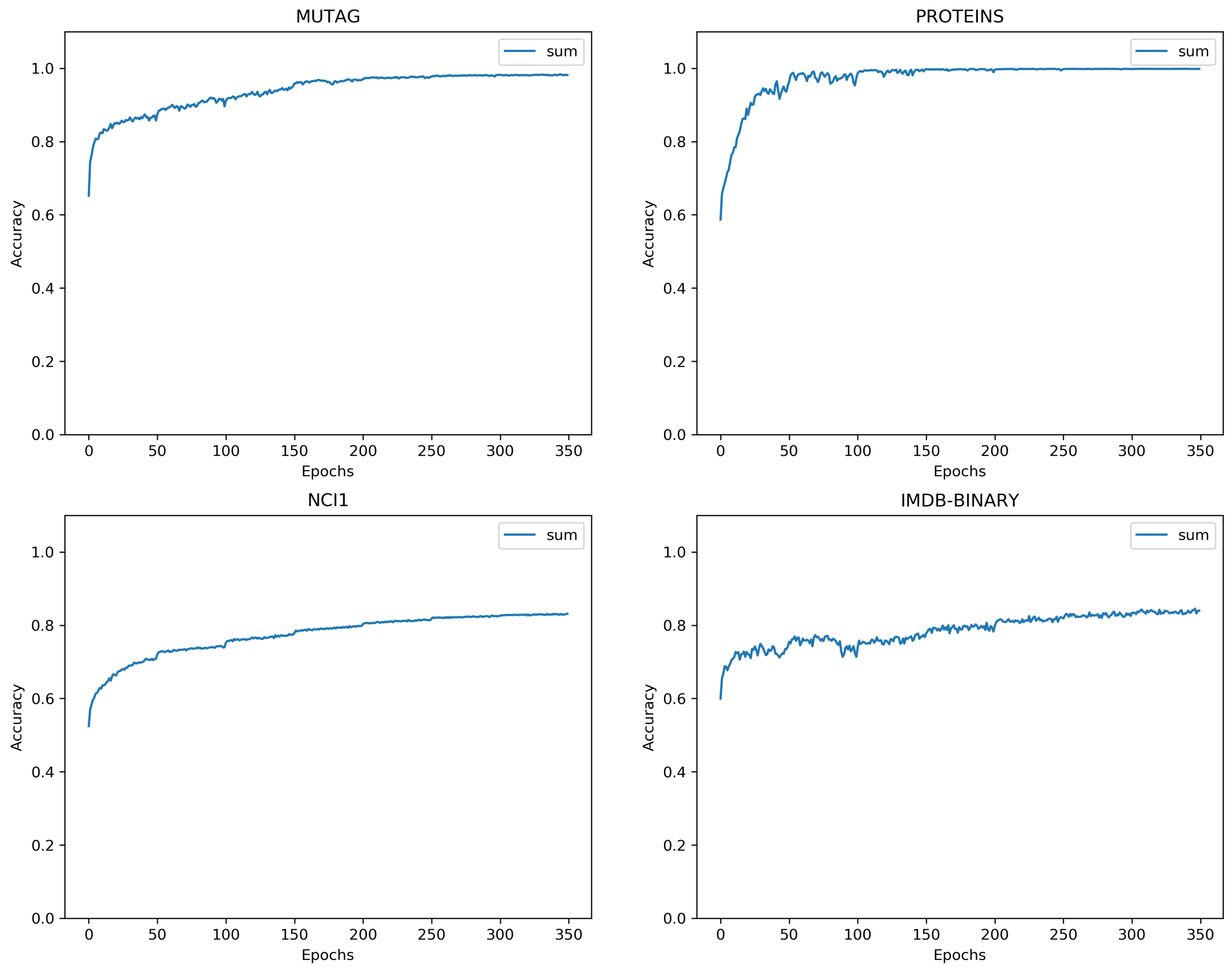


Figure 4: Average training accuracies (over 10 runs) of the GNN architecture described in section 2.5.1 but using a prime factorization AGGREGATE function.

Although the accuracy during training was lower than the original methods (see table 1), the performance at test time still could not be distinguished from the previous experiments (taking into account the standard deviation of the test accuracies).

Phase	MUTAG	PROTEINS	NCI1	IMDB-BINARY
Training	98.126±0.435	99.836±0.114	83.147±0.971	83.977±2.496
Test	87.228±4.951	64.790±4.363	77.764±2.577	72.600±3.720

Table 4: Average training and test accuracies (cross-validated over 10 runs) of the GNN architecture described in section 2.5.1 but using a prime factorization AGGREGATE function.

5 Future Work

The results presented in the previous section show that changes in the COMBINE function that preserve the injectivity do not seem to greatly modify the results at test time. Thus, learning an optimal COMBINE function while forcing the injectivity (which, as proved in the section 2, is a key property for being able to distinguish graphs) could be an interesting line of research. Furthermore, learning the AGGREGATE function could also be an advantage, since the summation proposed in section 2.5.1 only preserves the injectivity if the input is one-hot encoded, an attribute that cannot be guaranteed in the inner layers of the GNN.

References

[1] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," *CoRR*, vol. abs/1810.00826, 2018.

[2] J. Atwood and D. Towsley, "Search-convolutional neural networks," *CoRR*, vol. abs/1511.02136, 2015.

[3] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," *CoRR*, vol. abs/1605.05273, 2016.

[4] S. Verma and Z.-L. Zhang, "Graph Capsule Convolutional Neural Networks," *arXiv e-prints*, p. arXiv:1805.08090, May 2018.

[5] S. Ivanov and E. Burnaev, "Anonymous walk embeddings," *CoRR*, vol. abs/1805.11921, 2018.

[6] B. L. Douglas, "The Weisfeiler-Lehman Method and Graph Isomorphism Testing," *arXiv e-prints*, p. arXiv:1101.5211, Jan 2011.