Supplementary Material for “A Novel Higher-order Weisfeiler-Lehman Graph Convolution”

Clemens Damke CDAMKE@MAIL.UPB.DE
Vitalik Melnikov MELNIKOV@MAIL.UPB.DE
Eyke Hüllermeier EYKE@UPB.DE
Paderborn University
Heinz Nixdorf Institute and Department of Computer Science
Intelligent Systems and Machine Learning Group

Editors: Sinno Jialin Pan and Masashi Sugiyama

This supplement contains additional details about the 2-WL-GNN model that was proposed in the main paper. We begin with a description of how this model can be efficiently implemented on modern hardware. Afterwards, details about the evaluation procedure and the chosen datasets are provided. Finally, we discuss a few additional experimental results that were left out of the main paper.

Appendix A. Implementation of 2-WL-GNNs on GPGPUs

In the main paper we focused on the discriminative power of the proposed 2-WL-GNN. In this supplementary section we describe how our approach can be implemented on *general purpose graphics processing units* (GPGPUs).

Efficient high-level software libraries for the implementation of vertex neighborhood convolution approaches such as graph convolutional network or *graph isomorphism network* (GIN) already exist. They describe convolutions via a message-passing abstraction in which vertex feature vectors are passed along their neighboring edges (see Battaglia et al., 2018). Since a message-passing model along edges is incompatible with the edge-pair neighborhoods of 2-WL, a custom convolution implementation is required for 2-WL-GNNs.

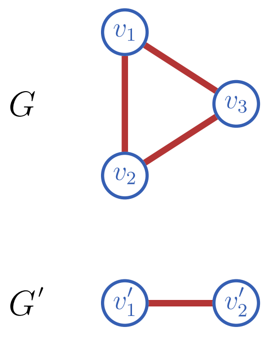
For this purpose we propose a sparse 2-WL graph representation which is inspired by the coordinate list adjacency format described by Fey and Lenssen (2019). Given a neighborhood radius r, we encode a graph G using the following two matrices:

1. $Z_G^{(0)} \in \mathbb{R}^{m \times d^{(0)}}$: The initial feature matrix is represented directly as a dense floating point matrix with $m := |\mathcal{E}_{G^r}|$ rows, each of which encodes the feature vector of an edge $e_{ij} \in \mathcal{E}_{G^r}$. Edge feature duplicates are prevented by only encoding edges with $i \leq j$ for some arbitrary vertex ordering of G.

2. $R_G \in [m]^{\gamma \times 3}$: The reference matrix R_G encodes the edge neighborhood information. It consists of $\gamma := \sum_{e_{ij} \in \mathcal{E}_{G^r}} |\Gamma_{G^r}(v_i) \cap \Gamma_{G^r}(v_j)|$ rows, one for each 2-WL neighbor $\{\!\{e_{il}, e_{lj}\}\!\}$ of each edge e_{ij}. Each neighbor row is a vector $(r_{\mathrm{L}}, r_{\Gamma,1}, r_{\Gamma,2}) \in [m]^3$ of three index pointers to rows in $Z_G^{(0)}$. r_{L} points to the row index of the feature vector of e_{ij},

while $r_{\Gamma,1}$ and $r_{\Gamma,2}$ point to the indices of e_{il} and e_{lj} respectively. We will refer to the three column vectors of R_G as $R_{G,\mathrm{L}}$, $R_{G,\Gamma,1}$ and $R_{G,\Gamma,2}$.

This encoding can also be used to represent graph batches by simply concatenating the rows of each graph's feature and reference matrices while shifting the index pointers to account for the concatenation offsets. Figure 1 illustrates how such a batch encoding might look like. After encoding a graph dataset as 2-WL matrices, convolutions can be computed efficiently



idx.	edge	$Z^{(0)}$
1	e_{11}	$(1,0)$
2	e_{22}	$(1,0)$
3	e_{33}	$(1,0)$
4	e_{12}	$(0,1)$
5	e_{13}	$(0,1)$
6	e_{23}	$(0,1)$
7	e'_{11}	$(1,0)$
8	e'_{22}	$(1,0)$
9	e'_{12}	$(0,1)$

R_{L}	$R_{\Gamma,1}$	$R_{\Gamma,2}$
(1,	1,	1)
(1,	4,	4)
(1,	5,	5)
(2,	2,	2)
(2,	4,	4)
(2,	6,	6)
(3,	3,	3)
(3,	5,	5)
(3,	6,	6)
(4,	1,	4)
(4,	4,	2)
(4,	5,	6)
(5,	1,	5)
(5,	5,	3)
(5,	4,	6)
(6,	2,	6)
(6,	6,	3)
(6,	4,	5)
(7,	7,	7)
(7,	9,	9)
(8,	8,	8)
(8,	9,	9)
(9,	7,	9)
(9,	9,	8)

Figure 1: Exemplary 2-WL encoding of a batch of two small graphs.

on GPGPUs via the common *gather-scatter* pattern from parallel programming (He et al., 2007). The so-called *gather* operator takes two inputs: A list Z of m row vectors and a list R of γ pointers into Z. It returns a list X of γ row vectors $X[i] = Z[R[i]]$ for $i \in [\gamma]$. The *scatter*$_\Sigma$ operator can be understood as the opposite of *gather*. *scatter*$_\Sigma$ takes a list X of γ row vectors and a list R of γ pointers from the range $[m]$. It returns a list Z of m row vectors $Z[i] = \sum_{j \in [\gamma] \wedge R[j] = i} X[j]$ for $i \in [m]$.

Using the *gather* and *scatter*$_\Sigma$ operators, the 2-WL convolution operator from Def. 4.3 can be computed via the following parallel algorithm:

Algorithm 1 Parallel Implementation of a 2-WL Convolution Layer $S^{(t)}$

```
1:  function $S^{(t)}(Z^{(t-1)} \in \mathbb{R}^{m \times d^{(t-1)}}, R \in [m]^{\gamma \times 3})$
2:      $Z_{\mathrm{L}} := Z^{(t-1)} W_{\mathrm{L}}^{(t)}$                          ▷ Matrix multiply: $\mathbb{R}^{m \times d^{(t-1)}} \to \mathbb{R}^{m \times d^{(t)}}$
3:      $Z_{\mathrm{F}} := Z^{(t-1)} W_{\mathrm{F}}^{(t)}$
4:      $Z_{\Gamma} := Z^{(t-1)} W_{\Gamma}^{(t)}$
5:      $X_{\Gamma,1} := gather(Z_{\Gamma}, R_{\Gamma,1})$                 ▷ Gather: $\mathbb{R}^{m \times d^{(t)}} \times [m]^{\gamma} \to \mathbb{R}^{\gamma \times d^{(t)}}$
6:      $X_{\Gamma,2} := gather(Z_{\Gamma}, R_{\Gamma,2})$
7:      $X_{\Gamma} := \sigma_{\Gamma}(X_{\Gamma,1} + X_{\Gamma,2})$                    ▷ Element-wise operations
8:      $Z_{\Sigma\Gamma} := scatter_{\Sigma}(X_{\Gamma}, R_{\mathrm{L}})$               ▷ Scatter: $\mathbb{R}^{\gamma \times d^{(t)}} \times [m]^{\gamma} \to \mathbb{R}^{m \times d^{(t)}}$
9:      $Z^{(t)} := \sigma(Z_{\mathrm{L}} + Z_{\mathrm{F}} \odot Z_{\Sigma\Gamma})$                  ▷ Element-wise operations
10:     return $Z^{(t)}$
```

Appendix B. Evaluated Hyperparameter Grids

To tune the hyperparameters of the evaluated models, we used a regular grid search. Depending on the type of model, different sets of hyperparameter configurations Θ were used.

Graph Kernels We used the support vector machine classifier from Scikit-learn to evaluate the graph kernel approaches. We tuned only the regularization parameter `C` of this classifier; the evaluated values are $\texttt{C} \in \{1, 1 \times 10^{-1}, 1 \times 10^{-2}, 1 \times 10^{-3}, 1 \times 10^{-4}\}$. All other parameters were left at the default setting (using `scikit-learn 0.22.1`).

Baseline and GIN For the evaluation of the structure unaware baseline learner and GIN, we used the same hyperparameter configurations as Errica et al. (2020). We therefore refer to their work for a complete list of the tuned hyperparameters for those models.

2-GNN and 2-WL-GNN We evaluated our implementations of 2-GNNs as well as 2-WL-GNNs on the grid spanned by the following hyperparameter values:

- **Number of convolutional layers $T \in \{3, 5\}$:** This parameter describes only the depth of the stack of convolutional layers. The *multilayer perceptron* (MLP) after the pooling layer is always configured with a single hidden layer.
- **Layer width $d \in \{32, 64\}$:** This parameter describes the output dimensionalities $d = d^{(1)} = \cdots = d^{(T)}$ of the convolutional layers and (if applicable) also the hidden layer width of the final MLP after the pooling layer.
- **Learning rate** $\eta \in \{1 \times 10^{-2}, 1 \times 10^{-3}, 1 \times 10^{-4}\}$ of the Adam optimizer.
- **Activation functions σ and σ_Γ** are set to the standard logistic function. However, for the evaluation of the synthetic TRIANGLE dataset we used ReLU instead because this choice led to improved and more consistent results in previous exploratory experiments.
- **Number of epochs E and early stopping patience p** are set to $E = 1000$ and $p = 100$, except for the evaluation of the synthetic TRIANGLE dataset for which we used $E = 5000$ and $p = 1000$ to ensure model convergence.

Both, 2-GNNs and 2-WL-GNNs, were evaluated using two different pooling layers which combine the edge feature vectors $\{z_{ij}\}_{e_{ij} \in \mathcal{E}_{G^r}}$ into a graph feature representation z_G. The mean pooling layer uses $z_G = \frac{1}{|\mathcal{E}_{G^r}|} \sum_{e_{ij}} z_{ij}$. The weighted mean pooling layer extends this approach by incorporating attention scores $w_{ij} \in \mathbb{R}$ that are learned alongside z_{ij} for each edge; the graph feature representation is then defined as $z_G = \frac{1}{\sum_{e_{ij}} e^{w_{ij}}} \sum_{e_{ij}} e^{w_{ij}} z_{ij}$.

Appendix C. Dataset Statistics and Descriptions

TRIANGLE The triangle detection dataset was generated by sampling three graphs with exactly one unicolored triangle uniformly at random for each possible combination of the following parameters: The number of vertices (between 6 and 32), the vertex color proportions (either 50/50%, 75/25% or 25/75% vertices with the colors **A**/**B**), the graph density ($|\mathcal{V}_G|^{-2} \, |\mathcal{E}_G| \in \{1/4, 1/2\}$) the graph class (add a triangle with either the color **A** or **B**).

Table 1: Sizes of the evaluated binary classification datasets and their graphs.

	no. of graphs	vertex data (feat. + lab.)	vertex count $\lvert\mathcal{V}_G\rvert$ min	mean	max	edge count $\lvert\mathcal{E}_G\rvert$ min	mean	max	vert. deg. mean $\pm$ σ
TRIANGLE	228	$0+2$	6	18.3	32	5	52.1	164	5.7 ± 2.8
NCI1	4110	$0+37$	3	29.9	111	2	32.3	119	2.2 ± 0.8
PROTEINS	1113	$29+3$	4	39.1	620	5	72.8	1049	3.7 ± 1.1
D&D	1178	$0+89$	30	284.3	5748	63	715.7	14267	5.0 ± 1.7
REDDIT	2000	$0+1$	6	429.6	3782	4	497.8	4071	2.3 ± 20.7
IMDB	1000	$0+1$	12	19.8	136	26	96.5	1249	9.8 ± 7.4

NCI1 This dataset was made available by Shervashidze et al. (2011). It contains a balanced subset of molecule graphs that were originally published by the US National Cancer Institute. In each molecule graph, vertices correspond to atoms and edges to bonds between them. The binary classes in this dataset describe whether a molecule is able to suppress or inhibit the growth of certain lung cancer and ovarian cancer cell lines in humans.

PROTEINS and D&D The graphs in both the PROTEINS dataset (Borgwardt et al., 2005) as well as the D&D dataset (Dobson and Doig, 2003) represent proteins. Each vertex corresponds to a so-called *secondary structure element* (SSE), i.e. a certain molecular substructure. An edge encodes either that two SSEs are neighbors in the protein's amino-acid sequence or that those SSEs are close to each other in 3D space. Each protein graph is classified by whether it is an enzyme or not. The main difference between the two datasets is their selection of vertex features/labels.

REDDIT This balanced dataset contains graphs that represent online discussion threads on the website Reddit (Yanardag and Vishwanathan, 2015). Each vertex corresponds to a user; an edge is drawn between two users iff. at least one of them replied to a comment of another. Such social interaction graphs were sampled from two types of subreddits: Question/answer-based and discussion-based. The classification goal is to predict from which type of subreddit a given graph was sampled.

IMDB This dataset contains so-called *ego-networks* of movie actors (Yanardag and Vishwanathan, 2015). Vertices in such networks represent actors and edges encode whether two actors starred in the same movie. The graphs in the dataset are derived from the actors starring in either action or romance movies. The classification goal for each graph is to predict the movie genre it was derived from.

Appendix D. Influence of the Neighborhood Radius on the Predictive Performance

As described in Section 4, the neighborhood radius $r \in \mathbb{N}$ determines the number of convolved edges feature vectors, i.e. the number of rows in $Z^{(t)} \in \mathbb{R}^{|\mathcal{E}_{G^r}| \times d^{(t)}}$ (see Fig. 2). We will now analyze the influence of r on the accuracy of 2-WL-GNNs. The theory suggests that the DP of the 2-WL convolution is increased by increasing the radius; e.g., the proof of 2-WL's cycle counting ability (Fürer, 2017) depends heavily on the structural information carried by the indirect edge features/colors that are only present if $r > 1$.

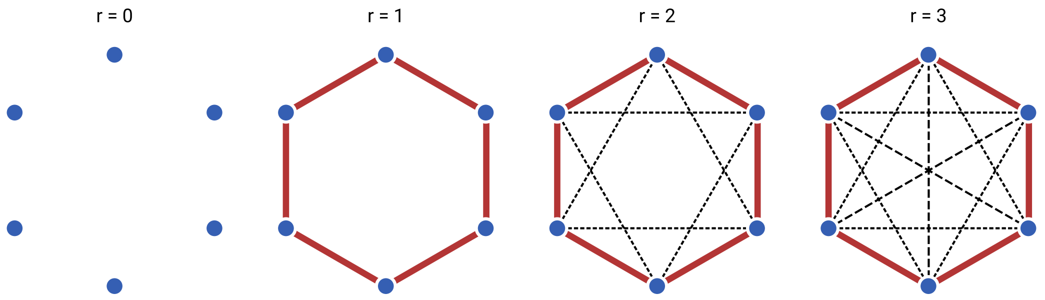


Figure 2: Illustration of the powers of the six-cycle graph for varying r. The self-loop edge at each of the vertices is not explicitly shown. For $r = 3$ all possible edges between the six vertices will be considered, just as in the original 2-WL algorithm.

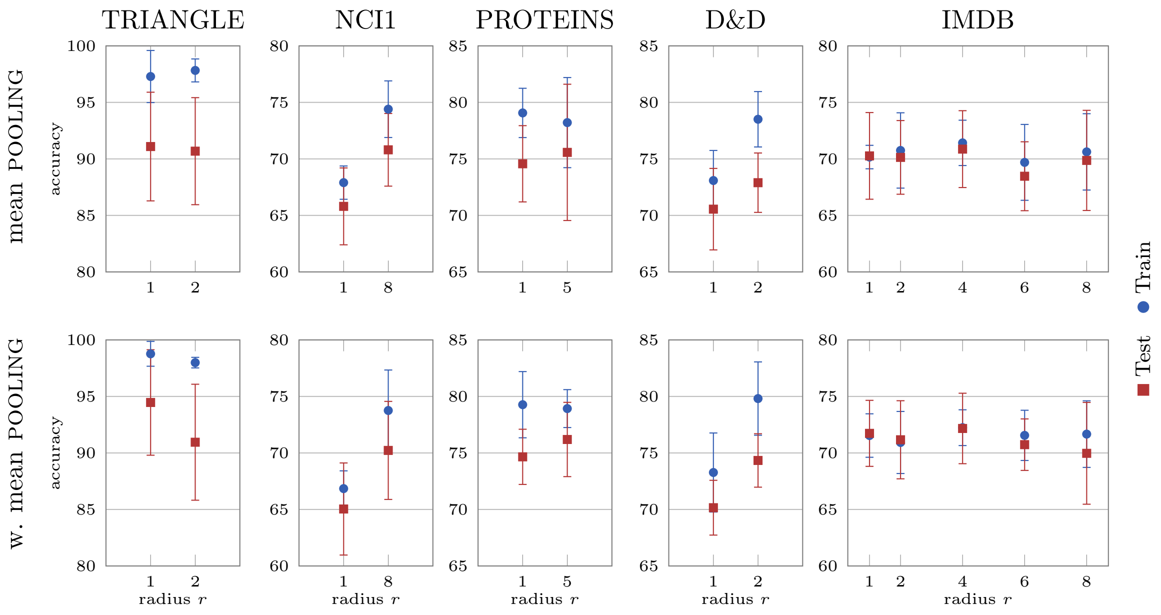


Figure 3: Accuracy of the 2-WL-GNN with varying neighborhood radii r. All datasets were evaluated on $r = 1$ and the highest radius for which the 2-WL graph encodings would still fit into memory; the REDDIT dataset only fit into memory for $r = 1$, therefore it is not shown here.

Figure 3 shows[1] that, in practice, a neighborhood radius $r > 1$ does correlate with a higher training and test accuracy on the NCI1 and D&D datasets; howerver, on the IMDB and PROTEINS datasets this is not the case. This difference is interesting because NCI1 (molecular structures) and D&D (protein sequences) contain more cyclic graphs, while IMDB (ego-network structures) and PROTEINS (protein sequences) consist of more tree- or list-like graphs (see Appendix C). Even though the PROTEINS and D&D datasets both contain

1. Note that the accuracies in this figure are lower than those in the evaluation section of the main paper because a different 2-WL-GNN architecture is used here; namely, no MLP is applied after pooling.

protein sequences, we find that the protein sequences in the PROTEINS dataset are very "list-like" with much fewer large cycles than in the proteins structures of the D&D dataset (compare the vertex and edge count statistics of both datasets in Tbl. 1). Figure 4 illustrates this difference. This leads us the the hypothesis that 2-WL-GNNs with a neighborhood radius of $r > 1$ are able to improve their real-world performance over that achieved with $r = 1$ by detecting cyclic constituents in graphs. Due to the limited number of evaluation results, further investigations are required to verify this hypothesis.

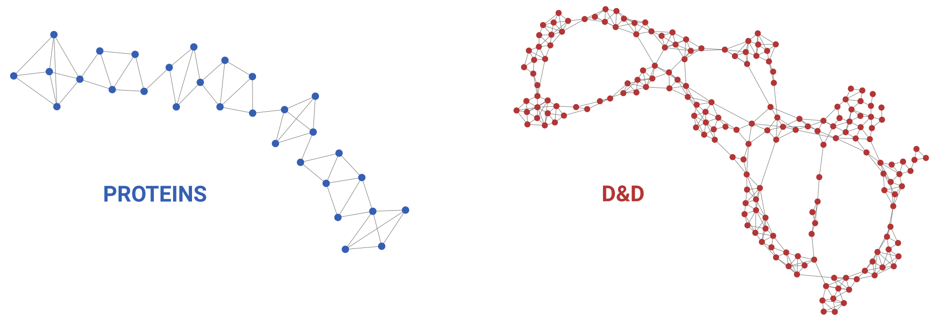
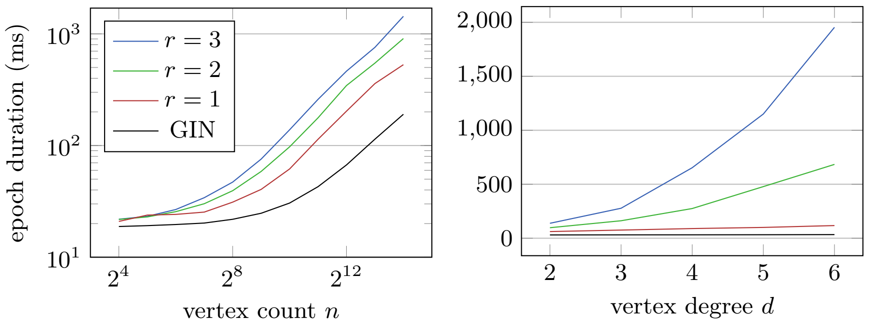


Figure 4: Two samples illustrating the difference between the PROTEINS and D&D datasets.

Appendix E. Empirical Runtime Evaluation



d	$r = 1$	$r = 2$	$r = 3$
2	2.0	3.2	4.5
3	2.4	5.2	8.9
4	2.8	8.7	20.7
5	3.1	14.8	35.6
6	3.5	20.4	58.2

Figure 5: Comparison of the mean training epoch durations of 2-WL-GNN and GIN for varying graph sizes n, vertex degrees d and neighborhood radii r. *(Left)* Durations for varying vertex counts $n \in \{2^4, \ldots, 2^{14}\}$ with a fixed degree $d = 2$. *(Middle)* Durations for varying vertex degrees d with a fixed size $n = 1024$. *(Right)* Table of the factors by which 2-WL-GNN is slower than GIN for $n = 1024$.

In Appendix D we evaluated the relation between the neighborhood radius r and the resulting accuracy. As described in Section 4, the neighborhood radius also affects the 2-WL convolution runtime which is bounded by $\mathcal{O}(nd^{2r})$, with n denoting the vertex count and d being the maximum vertex degree.

We will now verify this bound experimentally. Figure 5 shows the duration of a single training epoch of a 2-WL-GNN and a GIN model with the same depth and a similar number of learnable parameters (≈ 2200). The time for each combination of n, d and r was obtained by taking the mean duration of the first 100 training epochs using a dataset of 100 synthesized regular graphs with size n and uniform vertex degree d. The experiment was implemented in TensorFlow and executed on a single Nvidia 1080 Ti GPU. Since the coefficient of variation for the 100 samples of each (n, d, r) combination is $\leq 5\%$, no error bars are shown for visual clarity.

In the left plot of Fig. 5 we see that the epoch durations are dominated by constant costs for $n < 2^{10}$; for $n \geq 2^{10}$ the expected linearity in n can be observed. For 2-WL-GNNs we expect that the slope of the epoch durations is described by $\mathcal{O}(d^{2r})$. The roughly uniform y-offsets of the $r = 1$, $r = 2$ and $r = 3$ curves in the left log-log plot are in line with this expectation. Additionally, the middle plot confirms the expected power law relation between the epoch duration and the vertex degree d.

The table on the right shows how much slower the 2-WL-GNN is compared to the 1-WL bounded GIN architecture. Note however that we considered the worst-case scenario in which all vertices reach the upper degree bound d. Many real-world datasets consist of graphs with only a few high-degree vertices. Looking at the NCI1, PROTEINS, D&D, REDDIT and IMDB datasets, which have mean vertex degrees that range between 2.2 and 9.8 (see Tbl. 1), the table in Fig. 5 would suggest that a 2-WL-GNN with $r = 1$ is at least 2 or 3 times slower than a GIN on those datasets. However, in reality the slowdown is only $\frac{228\text{ms}}{198\text{ms}} \approx 1.15$ for NCI1, $\frac{262\text{ms}}{206\text{ms}} \approx 1.27$ for PROTEINS, $\frac{765\text{ms}}{514\text{ms}} \approx 1.49$ for D&D, $\frac{1081\text{ms}}{804\text{ms}} \approx 1.35$ for REDDIT and $\frac{96\text{ms}}{61\text{ms}} \approx 1.57$ for IMDB. For neighborhood radii $r > 1$, the real-world slowdown factor can be significantly smaller than the worst-case slowdown as well: $\frac{964\text{ms}}{198\text{ms}} \approx 4.87$ for NCI1 with $r = 8$, $\frac{670\text{ms}}{206\text{ms}} \approx 3.25$ for PROTEINS with $r = 5$, $\frac{1386\text{ms}}{514\text{ms}} \approx 2.70$ for D&D with $r = 2$ and $\frac{239\text{ms}}{61\text{ms}} \approx 3.92$ for IMDB with $r = 4$. Consequently, 2-WL-GNNs appear to be computationally feasible in practice despite the large worst-case slowdown factors in Fig. 5. Additionally, as we saw in Appendix D, a neighborhood radius $r > 1$ is not always necessary to reach optimal predictive performance; the computationally cheap choice of $r = 1$ should therefore always be considered.

Appendix F. Fold-wise Accuracy Deltas

Due to the relatively small sizes of the evaluated benchmark datasets, the variance of the test accuracies across different folds is quite large. When directly comparing the mean accuracies of two learners, it is often impossible to tell whether one consistently outperforms the other. We therefore now list the mean and standard deviations of the fold-wise test accuracy differences of all pairs of learners for all datasets. This effectively removes the variance introduced by "easy" and "hard" folds on which all learners might tend to perform consistently better/worse.

In the following Matrices 2 to 7 we show accuracy differences as *row accuracy* minus *column accuracy*. For each row i and column j, the corresponding cell (i, j) is highlighted in red or green iff. the learner i performs consistently worse (or better respectively) than j with a significance level of 2σ. To compute the deltas for 2-WL-GNNs, the same neighborhood radii as in the evaluation section of the main paper are used, i.e. $r = 2$ for the synthetic triangle detection dataset and $r = 8$, 5, 2, 1 and 4 for NCI1, PROTEINS, D&D, REDDIT and IMDB respectively.

Matrix 2: Fold-wise accuracy delta means and standard deviations on the triangle dataset.

	WL$_{\mathbf{ST}}$ ($T = 3$)	**WL**$_{\mathbf{SP}}$ ($T = 3$)	**2-LWL** ($T = 3$)	**2-GWL** ($T = 3$)	**Baseline** (sum)	**GIN** (sum)	**2-GNN** (mean)	**2-GNN** (w.m.)	**2-WL-GNN** (mean)	**2-WL-GNN** (w.m.)
WL$_{\mathbf{ST}}$ ($T = 3$)		−11 ±13	+0.4 ±9.6	−4.9 ±16	+12 ±12	−13 ±10	−20 ±14	−25 ±14	−36 ±17	−42 ±11
WL$_{\mathbf{SP}}$ ($T = 3$)	+11 ±13		+11 ±13	+6.1 ±9.2	+23 ±16	−2.0 ±14	−8.8 ±13	−14 ±14	−25 ±12	−31 ±11
2-LWL ($T = 3$)	−0.4 ±9.6	−11 ±13		−5.3 ±11	+12 ±7.5	−13 ±6.7	−20 ±13	−25 ±9.4	−36 ±10	−43 ±6.8
2-GWL ($T = 3$)	+4.9 ±16	−6.1 ±9.2	+5.3 ±11		+17 ±15	−8.2 ±13	−15 ±14	−20 ±13	−31 ±9.0	−38 ±9.0
Baseline (sum)	−12 ±12	−23 ±16	−12 ±7.5	−17 ±15		−25 ±8.4	−32 ±14	−37 ±9.2	−48 ±14	−55 ±8.5
GIN (sum)	+13 ±10	+2.0 ±14	+13 ±6.7	+8.2 ±13	+25 ±8.4		−6.8 ±13	−12 ±9.5	−23 ±14	−29 ±7.3
2-GNN (mean)	+20 ±14	+8.8 ±13	+20 ±13	+15 ±14	+32 ±14	+6.8 ±13		−5.0 ±6.6	−16 ±11	−23 ±9.8
2-GNN (w.m.)	+25 ±14	+14 ±14	+25 ±9.4	+20 ±13	+37 ±9.2	+12 ±9.5	+5.0 ±6.6		−11 ±8.8	−18 ±7.2
2-WL-GNN (mean)	+36 ±17	+25 ±12	+36 ±10	+31 ±9.0	+48 ±14	+23 ±14	+16 ±11	+11 ±8.8		−6.5 ±8.7
2-WL-GNN (w.m.)	+42 ±11	+31 ±11	+43 ±6.8	+38 ±9.0	+55 ±8.5	+29 ±7.3	+23 ±9.8	+18 ±7.2	+6.5 ±8.7	

Matrix 3: Fold-wise accuracy delta means and standard deviations on NCI1.

	WL$_{\mathbf{ST}}$ ($T = 1$)	**WL**$_{\mathbf{ST}}$ ($T = 3$)	**2-LWL** ($T = 3$)	**2-GWL** ($T = 3$)	**Baseline** (sum)	**GIN** (sum)	**2-GNN** (mean)	**2-GNN** (w.m.)	**2-WL-GNN** (mean)	**2-WL-GNN** (w.m.)
WL$_{\mathbf{ST}}$ ($T = 1$)		−11 ±1.3	−2.8 ±1.5	+2.3 ±2.4	+6.2 ±2.9	−3.5 ±2.2	−1.9 ±2.9	−4.4 ±2.3	+1.6 ±2.6	+0.4 ±2.1
WL$_{\mathbf{ST}}$ ($T = 3$)	+11 ±1.3		+8.1 ±1.6	+13 ±1.8	+17 ±2.9	+7.3 ±2.7	+8.9 ±2.3	+6.5 ±2.0	+12 ±2.5	+11 ±2.0
2-LWL ($T = 3$)	+2.8 ±1.5	−8.1 ±1.6		+5.1 ±2.3	+9.0 ±3.0	−0.7 ±2.7	+0.9 ±2.0	−1.6 ±2.3	+4.4 ±2.5	+3.2 ±2.5
2-GWL ($T = 3$)	−2.3 ±2.4	−13 ±1.8	−5.1 ±2.3		+3.9 ±4.1	−5.8 ±3.6	−4.3 ±2.5	−6.7 ±2.5	−0.8 ±2.7	−1.9 ±3.1
Baseline (sum)	−6.2 ±2.9	−17 ±2.9	−9.0 ±3.0	−3.9 ±4.1		−9.8 ±2.4	−8.2 ±2.9	−11 ±2.6	−4.7 ±3.1	−5.8 ±2.8
GIN (sum)	+3.5 ±2.2	−7.3 ±2.7	+0.7 ±2.7	+5.8 ±3.6	+9.8 ±2.4		+1.6 ±3.5	−0.9 ±2.2	+5.1 ±3.2	+3.9 ±3.3
2-GNN (mean)	+1.9 ±2.9	−8.9 ±2.3	−0.9 ±2.0	+4.3 ±2.5	+8.2 ±2.9	−1.6 ±3.5		−2.4 ±2.6	+3.5 ±2.2	+2.4 ±2.6
2-GNN (w.m.)	+4.4 ±2.3	−6.5 ±2.0	+1.6 ±2.3	+6.7 ±2.5	+11 ±2.6	+0.9 ±2.2	+2.4 ±2.6		+5.9 ±3.2	+4.8 ±3.2
2-WL-GNN (mean)	−1.6 ±2.6	−12 ±2.5	−4.4 ±2.5	+0.8 ±2.7	+4.7 ±3.1	−5.1 ±3.2	−3.5 ±2.2	−5.9 ±3.2		−1.1 ±2.0
2-WL-GNN (w.m.)	−0.4 ±2.1	−11 ±2.0	−3.2 ±2.5	+1.9 ±3.1	+5.8 ±2.8	−3.9 ±3.3	−2.4 ±2.6	−4.8 ±3.2	+1.1 ±2.0	

Matrix 4: Fold-wise accuracy delta means and standard deviations on PROTEINS.

	WL$_{\mathbf{ST}}$ ($T=3$)	**WL**$_{\mathbf{SP}}$ ($T=3$)	**2-LWL** ($T=3$)	**2-GWL** ($T=3$)	**Baseline** (sum)	**GIN** (sum)	**2-GNN** (mean)	**2-GNN** (w.m.)	**2-WL-GNN** (mean)	**2-WL-GNN** (w.m.)
WL$_{\mathbf{ST}}$ ($T=3$)		−0.1 ±2.8	+3.6 ±4.4	−0.1 ±3.3	−1.0 ±5.1	+1.3 ±2.1	−1.8 ±3.6	−0.7 ±3.4	−3.5 ±3.1	−2.3 ±3.7
WL$_{\mathbf{SP}}$ ($T=3$)	+0.1 ±2.8		+3.7 ±3.5	−0.0 ±3.6	−0.9 ±4.1	+1.4 ±3.0	−1.7 ±4.1	−0.6 ±4.0	−3.4 ±3.2	−2.2 ±3.7
2-LWL ($T=3$)	−3.6 ±4.4	−3.7 ±3.5		−3.7 ±4.6	−4.6 ±5.7	−2.3 ±4.0	−5.4 ±3.1	−4.3 ±4.6	−7.1 ±3.8	−5.9 ±3.4
2-GWL ($T=3$)	+0.1 ±3.3	+0.0 ±3.6	+3.7 ±4.6		−0.9 ±4.9	+1.4 ±2.4	−1.7 ±3.7	−0.6 ±3.1	−3.4 ±3.0	−2.2 ±3.9
Baseline (sum)	+1.0 ±5.1	+0.9 ±4.1	+4.6 ±5.7	+0.9 ±4.9		+2.3 ±4.1	−0.8 ±5.9	+0.3 ±6.3	−2.5 ±4.6	−1.3 ±5.1
GIN (sum)	−1.3 ±2.1	−1.4 ±3.0	+2.3 ±4.0	−1.4 ±2.4	−2.3 ±4.1		−3.1 ±3.0	−2.0 ±3.0	−4.8 ±2.2	−3.6 ±2.8
2-GNN (mean)	+1.8 ±3.6	+1.7 ±4.1	+5.4 ±3.1	+1.7 ±3.7	+0.8 ±5.9	+3.1 ±3.0		+1.1 ±2.6	−1.7 ±2.1	−0.5 ±2.7
2-GNN (w.m.)	+0.7 ±3.4	+0.6 ±4.0	+4.3 ±4.6	+0.6 ±3.1	−0.3 ±6.3	+2.0 ±3.0	−1.1 ±2.6		−2.8 ±2.2	−1.6 ±2.8
2-WL-GNN (mean)	+3.5 ±3.1	+3.4 ±3.2	+7.1 ±3.8	+3.4 ±3.0	+2.5 ±4.6	+4.8 ±2.2	+1.7 ±2.1	+2.8 ±2.2		+1.2 ±2.1
2-WL-GNN (w.m.)	+2.3 ±3.7	+2.2 ±3.7	+5.9 ±3.4	+2.2 ±3.9	+1.3 ±5.1	+3.6 ±2.8	+0.5 ±2.7	+1.6 ±2.8	−1.2 ±2.1	

Matrix 5: Fold-wise accuracy delta means and standard deviations on D&D.

	WL$_{\mathbf{ST}}$ ($T=1$)	**WL**$_{\mathbf{ST}}$ ($T=3$)	**2-LWL** ($T=3$)	**2-GWL** ($T=3$)	**Baseline** (sum)	**GIN** (sum)	**2-GNN** (mean)	**2-GNN** (w.m.)	**2-WL-GNN** (mean)	**2-WL-GNN** (w.m.)
WL$_{\mathbf{ST}}$ ($T=1$)		+0.2 ±1.6	+2.4 ±3.3	+2.6 ±3.6	+3.2 ±2.7	+3.8 ±2.8	+6.0 ±6.7	+9.3 ±6.4	+3.5 ±4.4	+4.3 ±4.0
WL$_{\mathbf{ST}}$ ($T=3$)	−0.2 ±1.6		+2.2 ±3.5	+2.5 ±3.5	+3.1 ±3.3	+3.6 ±3.2	+5.9 ±6.4	+9.1 ±6.4	+3.4 ±4.7	+4.1 ±4.2
2-LWL ($T=3$)	−2.4 ±3.3	−2.2 ±3.5		+0.3 ±1.4	+0.9 ±3.7	+1.4 ±4.7	+3.7 ±6.2	+6.9 ±5.7	+1.2 ±4.5	+1.9 ±4.3
2-GWL ($T=3$)	−2.6 ±3.6	−2.5 ±3.5	−0.3 ±1.4		+0.6 ±4.3	+1.2 ±5.2	+3.4 ±6.9	+6.7 ±6.3	+0.9 ±5.2	+1.7 ±4.8
Baseline (sum)	−3.2 ±2.7	−3.1 ±3.3	−0.9 ±3.7	−0.6 ±4.3		+0.5 ±2.3	+2.8 ±4.9	+6.1 ±4.7	+0.3 ±3.4	+1.0 ±3.2
GIN (sum)	−3.8 ±2.8	−3.6 ±3.2	−1.4 ±4.7	−1.2 ±5.2	−0.5 ±2.3		+2.2 ±5.2	+5.5 ±5.4	−0.3 ±3.6	+0.5 ±3.5
2-GNN (mean)	−6.0 ±6.7	−5.9 ±6.4	−3.7 ±6.2	−3.4 ±6.9	−2.8 ±4.9	−2.2 ±5.2		+3.3 ±4.1	−2.5 ±4.7	−1.7 ±5.0
2-GNN (w.m.)	−9.3 ±6.4	−9.1 ±6.4	−6.9 ±5.7	−6.7 ±6.3	−6.1 ±4.7	−5.5 ±5.4	−3.3 ±4.1		−5.8 ±2.8	−5.0 ±3.3
2-WL-GNN (mean)	−3.5 ±4.4	−3.4 ±4.7	−1.2 ±4.5	−0.9 ±5.2	−0.3 ±3.4	+0.3 ±3.6	+2.5 ±4.7	+5.8 ±2.8		+0.8 ±1.0
2-WL-GNN (w.m.)	−4.3 ±4.0	−4.1 ±4.2	−1.9 ±4.3	−1.7 ±4.8	−1.0 ±3.2	−0.5 ±3.5	+1.7 ±5.0	+5.0 ±3.3	−0.8 ±1.0	

Matrix 6: Fold-wise accuracy delta means and standard deviations on REDDIT.

	WL$_{\text{ST}}$ $(T=1)$	**WL$_{\text{ST}}$** $(T=3)$	**2-LWL** $(T=3)$	**2-GWL** $(T=3)$	**Baseline** (sum)	**GIN** (sum)	**2-WL-GNN** (mean)	**2-WL-GNN** (w.m.)
WL$_{\text{ST}}$ $(T=1)$		−1.8 ±1.8	+0.5 ±4.9	+0.8 ±3.6	+4.2 ±10	−11 ±6.9	−7.4 ±6.2	−13 ±3.5
WL$_{\text{ST}}$ $(T=3)$	+1.8 ±1.8		+2.3 ±4.7	+2.6 ±3.5	+5.9 ±11	−9.0 ±7.1	−5.7 ±6.8	−11 ±2.9
2-LWL $(T=3)$	−0.5 ±4.9	−2.3 ±4.7		+0.3 ±4.2	+3.7 ±11	−11 ±7.3	−7.9 ±5.5	−14 ±3.4
2-GWL $(T=3)$	−0.8 ±3.6	−2.6 ±3.5	−0.3 ±4.2		+3.3 ±11	−12 ±7.6	−8.2 ±6.2	−14 ±2.7
Baseline (sum)	−4.2 ±10	−5.9 ±11	−3.7 ±11	−3.3 ±11		−15 ±12	−12 ±13	−17 ±10
GIN (sum)	+11 ±6.9	+9.0 ±7.1	+11 ±7.3	+12 ±7.6	+15 ±12		+3.3 ±9.6	−2.4 ±7.0
2-WL-GNN (mean)	+7.4 ±6.2	+5.7 ±6.8	+7.9 ±5.5	+8.2 ±6.2	+12 ±13	−3.3 ±9.6		−5.7 ±6.5
2-WL-GNN (w.m.)	+13 ±3.5	+11 ±2.9	+14 ±3.4	+14 ±2.7	+17 ±10	+2.4 ±7.0	+5.7 ±6.5	

Matrix 7: Fold-wise accuracy delta means and standard deviations on IMDB.

	WL$_{\text{ST}}$ $(T=3)$	**WL$_{\text{SP}}$** $(T=3)$	**2-LWL** $(T=3)$	**2-GWL** $(T=3)$	**Baseline** (sum)	**GIN** (sum)	**2-GNN** (mean)	**2-GNN** (w.m.)	**2-WL-GNN** (mean)	**2-WL-GNN** (w.m.)
WL$_{\text{ST}}$ $(T=3)$		−1.5 ±4.3	+0.7 ±2.0	+2.5 ±3.4	+22 ±4.9	+6.1 ±6.4	+1.5 ±2.6	+2.0 ±2.7	+1.7 ±3.7	+1.8 ±3.8
WL$_{\text{SP}}$ $(T=3)$	+1.5 ±4.3		+2.2 ±5.0	+4.0 ±5.2	+24 ±5.9	+7.6 ±7.4	+3.0 ±5.2	+3.5 ±5.7	+3.2 ±6.4	+3.3 ±6.7
2-LWL $(T=3)$	−0.7 ±2.0	−2.2 ±5.0		+1.8 ±2.6	+22 ±5.7	+5.4 ±7.2	+0.8 ±2.0	+1.3 ±3.2	+1.0 ±3.5	+1.1 ±3.7
2-GWL $(T=3)$	−2.5 ±3.4	−4.0 ±5.2	−1.8 ±2.6		+20 ±5.6	+3.6 ±7.1	−1.0 ±2.6	−0.5 ±4.1	−0.8 ±3.1	−0.7 ±4.3
Baseline (sum)	−22 ±4.9	−24 ±5.9	−21 ±5.7	−20 ±5.6		−16 ±6.3	−21 ±6.0	−20 ±5.6	−21 ±6.4	−20 ±6.9
GIN (sum)	−6.1 ±6.4	−7.6 ±7.4	−5.4 ±7.2	−3.6 ±7.1	+16 ±6.3		−4.6 ±7.5	−4.1 ±7.1	−4.4 ±7.9	−4.3 ±8.4
2-GNN (mean)	−1.5 ±2.6	−3.0 ±5.2	−0.8 ±2.0	+1.0 ±2.6	+21 ±6.0	+4.6 ±7.5		+0.5 ±2.8	+0.2 ±2.8	+0.2 ±2.4
2-GNN (w.m.)	−2.0 ±2.7	−3.5 ±5.7	−1.3 ±3.2	+0.5 ±4.1	+20 ±5.6	+4.1 ±7.1	−0.5 ±2.8		−0.3 ±3.5	−0.2 ±2.7
2-WL-GNN (mean)	−1.7 ±3.7	−3.2 ±6.4	−1.0 ±3.5	+0.8 ±3.1	+21 ±6.4	+4.4 ±7.9	−0.2 ±2.8	+0.3 ±3.5		+0.1 ±3.4
2-WL-GNN (w.m.)	−1.8 ±3.8	−3.3 ±6.7	−1.1 ±3.7	+0.7 ±4.3	+20 ±6.9	+4.3 ±8.4	−0.2 ±2.4	+0.2 ±2.7	−0.1 ±3.4	

References

P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu. Relational inductive biases, deep learning, and graph networks, 2018.

K. M. Borgwardt, C. S. Ong, S. Schonauer, S. V. N. Vishwanathan, A. J. Smola, and H.-P. Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21:i47–i56, 2005.

P. D. Dobson and A. J. Doig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of Molecular Biology*, 330(4):771–783, 2003.

F. Errica, M. Podda, D. Bacciu, and A. Micheli. A fair comparison of graph neural networks for graph classification. In *Int. Conf. on Learn. Rep.*, ICLR, 2020.

M. Fey and J. E. Lenssen. Fast graph representation learning with pytorch geometric, 2019.

M. Fürer. On the Combinatorial Power of the Weisfeiler-Lehman Algorithm. In *Lecture Notes in Computer Science*, pages 260–271. Springer International Publishing, 2017.

B. He, N. K. Govindaraju, Q. Luo, and B. Smith. Efficient gather and scatter operations on graphics processors. In *Proc. of the 2007 ACM/IEEE Conf. on Supercomputing*, 2007.

N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt. Weisfeiler-Lehman Graph Kernels. *JMLR*, 12:2539–2561, 2011.

P. Yanardag and S.V.N. Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015.