ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

3-WL GNNs for Metric Learning on Graphs

Aldo Moscatelli, Maxime Bérar,
Pierre Héroux, Florian Yger and Sébastien Adam.

Univ Rouen Normandie, INSA Rouen Normandie, Normandie Univ
LITIS UR4108, F-76000 Rouen, France

Abstract. Since the advent of Graph Neural Networks (GNNs), many works have computed distances between graphs by embedding them in vector spaces using Message Passing GNNs (MPNNs). However, MPNNs are known for their lack of expressiveness as they are bounded by the first-order Weisfeiler-Lehman test. In this paper, we use higher-order GNNs to tackle the metric learning problem and show on benchmark datasets how they can improve performance by using a node-level strategy and the Wasserstein distance.

1 Introduction

A key challenge in modeling structured information with graphs lies in computing the distances between them. The Graph Edit Distance(GED)[4] is a state-of-the-art method for this purpose; however, it suffers from NP-hard complexity. Recently, several architectures have been proposed to address this limitation [9, 7, 8, 11, 12] in a learning framework. These architectures generally consist of two main components. The first is an embedding block that uses siamese Graph Neural Networks(GNNs) to embed graphs either at the graph level or at the node level. The second component is a metric block that takes the embeddings generated by the first block as input and computes the distance between graphs, taking into account the embedding level. The rationale behind these architectures is that the embedding block learns an optimal representation to facilitate the computations in the metric block.

To the best of our knowledge, existing embedding blocks in the literature rely on simple yet effective Message Passing Neural Networks(MPNNs), such as GCN [3] or GIN [2]. Consequently, they suffer from the well-known limitations of MPNNs, including over-smoothing, over-squashing, and limited expressive power. This last limitation is particularly significant for metric learning, as it affects the ability to generate distinct embeddings for different graphs. Yet, GCN and GIN models have been shown to be at most equivalent to the first-order Weisfeiler-Lehman(WL) test in the WL hierarchy [1].

Recently, more expressive GNNs such as PPGN [5] and G^2N^2 [6] have been introduced in the literature, achieving a 3-WL expressivity level. To attain this level of expressivity, these architectures naturally incorporate edge embeddings, adding valuable information to the traditional node- and graph-level representations. These recent developments raise two research questions: how can 3-WL GNNs be integrated into a metric learning framework, and do they enable improved performance?

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

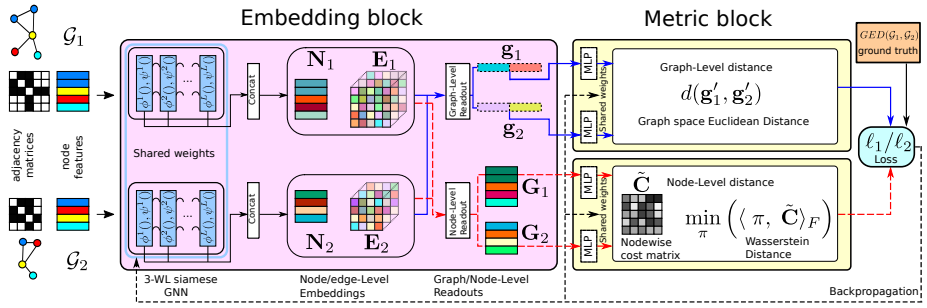

Fig. 1: Overview of the 3-WL graph metric learning framework. The solid blue arrow corresponds to the graph-level distance and the red dotted arrow corresponds to the node-level distance.

In this paper, we address these questions by investigating the impact of incorporating more expressive GNNs into Siamese-based metric learning architectures. Given that these GNNs produce richer outputs, specifically through edge-level embeddings for each graph, we propose two main strategies for computing the distance between graph pairs: a graph-level distance and a node-level matching approach using the Wasserstein distance. We conduct extensive experiments on benchmark datasets and demonstrate experimentally that the 3-WL expressivity, combined with a node-level distance strategy, enhances performance in metric learning tasks.

2 Expressive Graph Metric Learning

In order to examine the impact of more expressive GNNs within siamese-based metric learning architectures, we introduce in this paper the framework shown in Fig.1. It consists of two main blocks. The first block is the embedding block. It takes as input the adjacency tensors of both graphs and the node feature matrices containing the graph signals. This block can produce either a graph level or a node level embedding of both graphs (see Sec. 2.2). The second block, which is the metric block, takes as input the graph embeddings and computes a distance between these embeddings (see Sec. 2.3). Finally, a classical ℓ_1 or ℓ_2 regression loss is used to train the entire pipeline.

2.1 Inputs

The input of the siamese architecture is a pair of graphs. Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two graphs with n and m nodes, respectively, and let A_1 and A_2 be their adjacency tensors of shapes $(n \times n \times e)$ and $(m \times m \times e)$, where e denotes the edge feature dimension. F_1 and F_2 are two matrices of shapes $(m \times f)$ and $(n \times f)$ containing the node features.

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

2.2 Embedding block

The embedding block consists of two components. The first is a sequence of 3-WL layers (G^2N^2 or PPGN). It provides embeddings of edges and vertices at each layer through parameterised functions ϕ^l and ψ^l :

$$\mathcal{E}^{l+1} = \phi^l(\mathcal{E}^l, \mathcal{N}^l), \quad \mathcal{E}^0 = A, \quad \mathcal{N}^{l+1} = \psi^l(\mathcal{N}^l, \mathcal{E}^l), \quad \mathcal{N}^0 = F, \tag{1}$$

where the matrix $\mathcal{N}^l$ and the tensor $\mathcal{E}^l$ represent, respectively, the embeddings of the nodes and edges at each layer l. These representations obtained at each of the L layers of the GNNs are then concatenated :

$$\mathbf{E} = concat(\mathcal{E}^1, ..., \mathcal{E}^L), \quad \mathbf{N} = concat(\mathcal{N}^1, ..., \mathcal{N}^L). \tag{2}$$

The second component involves computing either a graph-level or node-level final embedding by applying an invariant or equivariant readout function. In the case of graph-level embedding, we use the classical graph-level readout function of 3-WL architectures:

$$\mathbf{g} = readout(\mathbf{N}, \mathbf{E}) = concat\left(sum(\mathbf{N}), sum(\mathbf{E} \odot \mathbf{I}), sum(\mathbf{E} \odot \mathbf{J})\right), \tag{3}$$

where $\mathbf{I}$ is the identity matrix and $\mathbf{J}$ is the off-diagonal matrix of ones, both of size equal to the number of graph nodes. Please note that the *sum* function is replaced by the *max* function for PPGN.

When considering node-level embedding, we propose in this paper the following equivariant readout function:

$$\mathbf{G} = \mathbf{N} + \sum_k \mathbf{E}_{:,k,:} \quad \text{with} \quad \mathbf{G}_{i,:} = \mathbf{N}_{i,:} + \sum_k \mathbf{E}_{i,k,:} \tag{4}$$

Using this equation, each row $\mathbf{G}_{i,:}$ represents a node embedding that also captures the structure centered on that node. It describes the structural information by aggregating the edge embedding information related to each node ($\mathbf{E}_{i,k,:}$) with the node representation itself ($\mathbf{N}_{i,:}$).

2.3 Metric block

As shown in Figure 1, two different strategies are used in the metric block according to the embedding level.

2.3.1 Graph-level distance strategy

The first strategy computes a graph-level distance between the embeddings of both graphs. In a first step, a siamese MLP embed $\mathbf{g}_1$ and $\mathbf{g}_2$ into a metric space $\mathbf{g}' = MLP(\mathbf{g})$. Then the graph distance $\mathcal{D}_{graph}$ is computed using the Euclidean distance d between the MLP outputs.

$$\mathcal{D}_{graph}(\mathcal{G}_1, \mathcal{G}_2) = d\left(\mathbf{g}'_1, \mathbf{g}'_2\right) \tag{5}$$

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

2.3.2 Node-level distance strategy

The second strategy should compute a node-level distance $\mathcal{D}_{node}(\mathcal{G}_1, \mathcal{G}_2)$ between node embeddings. Since each set of node embeddings can be considered as a discrete distribution, the Wasserstein distance can be used as in [8]. Considering two discrete distributions $\mu = \{(x_i, p_i)\}_{i=1}^{n}$ and $\nu = \{(y_j, q_j)\}_{j=1}^{m}$ with $x_i, y_j \in \mathbb{R}^n$ and p,q their probability distributions, the Wasserstein distance is defined as :

$$\mathcal{W}(\mu, \nu) = \min_{\pi \in \Pi} \left(\langle \pi,\ \mathbf{C} \rangle_F \right), \quad \text{with} \quad \Pi = \left\{ \pi, \pi \mathbb{1} = p, \pi^T \mathbb{1} = q \right\}, \tag{6}$$

where $\mathbf{C}$ is a matrix containing all pairwise dissimilarities, $\mathbf{C}_{i,j} = \mathcal{D}(x_i, y_j)$. In the following, we assume that the probability distributions are uniform.

In our case, the node-wise disimilarity matrix $\mathbf{C}$ is computed using the pairwise Euclidean distance between node embeddings $\mathbf{G}'_1$ and $\mathbf{G}'_2$:

$$\mathbf{C} = \begin{pmatrix} \mathbf{c}_{1,1} & \cdots & \mathbf{c}_{1,m} \\ \vdots & \ddots & \vdots \\ \mathbf{c}_{n,1} & \cdots & \mathbf{c}_{n,m} \end{pmatrix}, \mathbf{c}_{i,j} = \|(\mathbf{G}'_1)_{i,:} - (\mathbf{G}'_2)_{j,:}\|_2^2 \tag{7}$$

where $\mathbf{G}' = MLP(\mathbf{G})$ embeds the nodes representation into a metric space.

Since the graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ can have a different number of nodes (i.e. $n \neq m$), we use the classical strategy [10], which consists in extending the matrix $\mathbf{C}$ with dummy nodes equal to zero on its smallest dimension, resulting in $p = q$. In the following $n > m$ has been arbitrarily chosen defining a new cost matrix $\tilde{\mathbf{C}}$:

$$\tilde{\mathbf{C}} = \begin{pmatrix} \mathbf{C} & \mathbf{D} \end{pmatrix} \quad \text{with} \quad \mathbf{D} = \begin{pmatrix} \mathbf{d}_{1,1} & \cdots & \mathbf{d}_{1,n-m} \\ \vdots & \ddots & \vdots \\ \mathbf{d}_{n,1} & \cdots & \mathbf{d}_{n,n-m} \end{pmatrix}, \mathbf{d}_{i,j} = \|(\mathbf{G}'_1)_{i,:}\|_2^2 \tag{8}$$

Then, the node-level distance is provided by the value of the Wasserstein distance applied to $\tilde{\mathbf{C}}$ as defined in Eq. 6:

$$\mathcal{D}_{node}(\mathcal{G}_1, \mathcal{G}_2) = \mathcal{W}(\mathbf{G}'_1, \mathbf{G}'_2) = \min_{\pi} \left(\langle \pi,\ \tilde{\mathbf{C}} \rangle_F \right) \tag{9}$$

2.4 Model training

In our context of metric learning on graphs, the classical way to learn distances is to define a regression task on the ground truth distances of the datasets. The learning weights associated with ψ, ϕ and MLP are updated by backpropagation of either the ℓ_1 or the ℓ_2 loss functions.

3 Experiments and results

To showcase the performance of our approach, we conducted metric learning experiments based on the GED for three datasets: AIDS and IMDB-MULTI

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

		Linux			AIDS			IMDB		
Distance strategy	Architecture	GED Metrics MAE	GED Metrics MSE	Time s/100p	GED Metrics MAE	GED Metrics MSE	Time s/100p	GED Metrics MAE	GED Metrics MSE	Time s/100p
Graph-level	GREED	0.318	**0.172**	0.007	**0.629**	**0.634**	0.005	3.612	**45.347**	0.006
	G^2N^2(ours)	**0.284**	0.213	0.048	0.639	0.663	0.048	**3.459**	45.584	0.061
	PPGN(ours)	0.286	0.221	0.016	0.704	0.806	0.017	3.758	52.187	0.058
Node-level	SimGNN	0.489	0.443	0.244	0.816	1.075	0.283	28.082	4389.06	0.258
	GotSIM	-	0.329	-	-	0.992	-	-	4424.9	-
	GEDGNN	0.094	-	0.380	0.773	-	0.408	-	-	-
	GNOME	0.214	0.104	0.2653	0.555	0.490	0.265	2.976	**33.433**	0.272
	G^2N^2(ours)	0.071	**0.027**	0.095	**0.434**	**0.297**	0.092	**2.670**	38.43	0.147
	PPGN(ours)	**0.059**	0.028	0.086	0.443	0.316	0.066	2.727	36.238	0.113

Table 1: GED benchmarks evaluation. Methods are grouped according to their distance strategy. GED metrics are expressed in terms of MSE and MAE (lower is better, best results in **bold**). Inference times for 100 pairs are given in seconds.

from TUdataset [13] and Linux [14]. The GED ground truth of these datasets is provided by [9].

Table 1 shows the results obtained by our framework instantiated at node and graph levels with G^2N^2 and PPGN models against several methods from the literature [9, 7, 8, 11, 12]. The ℓ_1 loss is used for training for AIDS and Linux and the ℓ_2 loss for IMDB. Training is done with 200 epochs and batches of 32 graph pairs. The POT library[15] is used to compute the Wasserstein distance.

As shown in Table 1, the node-level strategy outperforms the graph-level strategy in terms of GED metrics, at the cost of a larger time consumption. With respect to other methods, our framework with either PPGN or G^2N^2 outperforms all other methods for node-level distance but shows no improvement for graph-level distance.

For the graph-level distance: since the datasets do not contain any pair of 1-WL equivalent non-isomorphic graphs[1], 1-WL (GIN and GCN) and 3-WL (PPGN and G^2N^2) GNNs are both able to separate all the pairs of the datasets at the graph-level. The difference in the produced graph embeddings does not allow a better learning of the GED metric for the 3-WL GNNs. It may be that all the information extracted by the 3-WL GNN is attenuated by the readout function and the distance calculation.

However, for the node-level distance, Table 1 experimentally shows a large performance improvement. As the distance strategy readout keeps the more expressive embeddings that describe nodes and their structural context information separated, the greater expressivity of the 3-WL GNNs can fully operate. Indeed these GNNs show good properties in shape detection and counting, such as triangles, squares, cycles up to length seven, paths up to length six, and various other shapes[6]. This is a desirable property when it comes to computing graph distance. Indeed similarity between graphs can be seen as the number of common shapes shared by the two graphs.

[1]We have performed the 1-WL test on all pairs of the datasets.

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

4 Conclusion and future work

In this paper we have proposed a two-block architecture that computes a graph metric value. The first block produces embeddings of both graphs by a 3-WL siamese GNN. The second block uses the Wasserstein distance to compute the dissimilarity between these graphs. We have conducted experiments that show the interest of model expressivity for metric learning between graphs when using a node-level distance strategy. The results obtained outperform state-of-the-art approaches and show that the node-level strategy gives better results than the graph-level strategy for metric learning between graphs. Future work will include the use of edge embeddings directly in the distance computation to achieve finer matching using the Fused Gromov-Wasserstein distance, another well-known optimal transport distance.

Acknowledgments and reproducibility: This work has been supported by the ANR-20-THIA-0021 and ANR-21- CE23-0025 grants. The source code of this work is available at https://github.com/aldomos/3WLMLG.

References

[1] B. Weisfeiler and A. Lehman. The reduction of a graph to canonical form and the algebra which appears therein, *Nauchno-Technicheskaya Informatsia, 1968.*

[2] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? ICLR*, 2019.*

[3] T. N. Kipf and M. Welling, Semi-supervised classification with graph convolutional networks, ICLR*, 2017.*

[4] H. Bunke and G. Allermann, Inexact graph matching for structural pattern recognition, PRL*, 1983.*

[5] H. Maron, H. Ben-Hamu et al. Provably Powerful Graph Networks, NeurIPS*, 2019.*

[6] J. Piquenot, A. Moscatelli, M. Berar, P. Héroux, R. Raveaux, J.-Y. Ramel and S. Adam, G^2N^2 *: Weisfeiler and Lehman go grammatical,* ICLR*, 2024.*

[7] C. Piao, T. Xu, X. Sun, Y.Rong, K. Zhao, and H. Cheng, Computing graph edit distance via neural graph matching, VLDB*, 2023.*

[8] K. D. Doan, S. Manchanda, et al. Interpretable graph similarity computation via differentiable optimal alignment of node embeddings, Conf. on Research and Development in Information Retrieval*, 2021.*

[9] Y. Bai, H. Ding, S. Bian, T. Chen, Y. Sun, and W. Wang, SIMGNN: A neural network approach to fast graph similarity computation, WSDM*, 2019.*

[10] S. Bougleux, L. Brun, V. Carletti, P. Foggia,B. Gauzere and M. Vento, Graph edit distance as a quadratic assignment problem, PRL*, 2017.*

[11] R. Ranjan, S. Grover, et al., GREED: A neural framework for learning graph distance functions, NeurIPS*, 2022.*

[12] A. Moscatelli, J.Piquenot et al., Graph node matching for edit distance, PRL*, 2024.*

[13] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann, Tu-dataset: A collection of benchmark datasets for learning with graphs, ICML Workshop on Graph Representation Learning and Beyond*, 2020.*

[14] X. Wang, X. Ding, A. K.H. Tung, S. Ying, and H. Jin, An efficient graph indexing method, Int. Conf. on Data Engineering*, 2012.*

[15] R. Flamary, N. Courty et al. POT: Python Optimal Transport, JMLR*, 2021.*