Matching-Graphs for Building Classification Ensembles

Mathias Fuchs[1(✉)] and Kaspar Riesen[1,2]

[1] Institute of Computer Science, University of Bern, 3012 Bern, Switzerland
{mathias.fuchs,kaspar.riesen}@unibe.ch
[2] Institute for Informations Systems, University of Applied Science and Arts Northwestern Switzerland, 4600 Olten, Switzerland
kaspar.riesen@fhnw.ch

Abstract. Ensemble learning is a well known paradigm, which combines multiple classification models to make a final prediction. Ensemble learning often demonstrates significant benefits, in particular a better classification performance than the individual ensemble members. However, in order to work properly, ensemble methods require a certain diversity of its members. One way to increase this diversity is to randomly select a subset of the available data for each classifier during the training process (known as bagging). In the present paper we propose a novel graph-based bagging ensemble that consists of graph neural networks. The novelty of our approach is that the ensemble operates on substantially augmented graph sets. The graph augmentation technique, in turn, is based on so-called matching-graphs, which can be computed on arbitrary pairs of graphs. In an experimental evaluation on five graph data sets, we show that this novel augmentation technique paired with a bagging ensemble is able to significantly improve the classification accuracy of several reference systems.

Keywords: Graph Matching · Matching-Graphs · Graph Edit Distance · Graph Augmentation · Graph Neural Network · Ensemble Learning

1 Introduction

Graphs, which consist of nodes that might be connected by edges, are used in a wide range of applications [1]. Indeed, graphs offer a compelling alternative to vector-based approaches, especially for applications involving complex data. This is mainly because graphs are capable of encoding more information than just an ordered and fixed-size list of real numbers.

In the last four decades a large number of procedures for graph-based pattern recognition has been proposed in the literature [2]. Those procedures range from graph edit distance [3], over spectral methods [4], to graph kernels [5] (to name just three examples). Recently, with the advent of Graph Neural Networks (GNNs) [6], the power of (deep) neural networks can finally be utilized by graphs.

Supported by Swiss National Science Foundation (SNSF) Project Nr. 200021_188496.

M. Vento et al. (Eds.): GbRPR 2023, LNCS 14121, pp. 102–112, 2023.
https://doi.org/10.1007/978-3-031-42795-4_10

In the present paper, we propose to use an ensemble learning method based on individual GNNs. In order to produce performant and robust GNNs, a lot of training data is typically required. Furthermore, it is accepted that ensemble learning methods perform the best when a large diversity of the individual classifiers is given [7]. The major contribution of the present paper is a novel method to generate both large and heterogeneous sets of graph data (particularly suited for ensemble learning). The novel method is based on a recently introduced data structure (known as *matching-graph* [8]).

Roughly speaking, a matching-graph encodes the matching subgraphs of two graphs under consideration. This basic concept can be used in many different ways. Matching-graphs can, for instance, improve the quality of graph dissimilarity computations by aggregating a matching-graph based distance and the original distance [8]. They can also be used to produce a subgraph based vector space embedding, by checking whether or not a set of given matching-graphs occur in the graph to embed [8]. The framework of matching-graphs is also successfully adopted for the automatic detection of relevant (i.e., frequent) substructures in very large graph sets [9]. Lastly, matching-graphs are also employed for graph augmentation in order to even out very small graph data sets [10], as well as to build more stable GNNs [11].

In the present work, we propose to further optimize the augmentation method presented in [11] to generate even more diverse matching-graphs. These novel matching-graphs provide a natural way to create realistic, diverse and relevant graphs of a specific class. It is our main hypothesis that the large amount of possible matching-graphs in conjunction with a bagging procedure ensures the diversity of the individual classifiers and finally allows to build a robust ensemble.

The remainder of this paper is organized as follows. In Sect. 2, we formally introduce the concept of matching-graphs and show how they can be used to augment a given training set of graphs and build a bagging ensemble. Eventually, in Sect. 3, we conduct an exhaustive experimental evaluation to provide empirical evidence that this novel approach is able to improve the classification accuracy of diverse reference systems. Finally, in Sect. 4, we conclude the paper and discuss potential ideas for future work.

2 Building an Ensemble with Matching-Graphs

Ensemble methods aim at combining several individual classifiers into one system. That is, an ensemble weighs the opinions of its individual members and combines their results to get the final decision [7]. Various ensemble methods have been proposed in the literature (e.g. Boosting [12] or Bagging [13]). In the present paper we employ – in principle – a bagging ensemble for graph classification. Thereby, the ensemble consists of multiple GNN classifiers that are trained on random subsets of the training data. The main contribution is to substantially increase the diversity of the bagging ensemble by means of matching-graphs. For this reason, we first introduce this basic concept (Subsect. 2.1) and then explain how these matching-graphs can be used for bagging (Subsect. 2.2).

2.1 Matching-Graphs

In the present paper, we use the following formalism to define graphs. A *graph* g is a four-tuple $g = (V, E, \mu, \nu)$, where V is the finite set of nodes, $E \subseteq V \times V$ is the set of edges, $\mu : V \rightarrow L_V$ is the node labeling function, and $\nu : E \rightarrow L_E$ is the edge labeling function.

Intuitively speaking, a matching-graph is built by extracting information about the matching of pairs of graphs and formalising this information into a new graph [8]. Matching-graphs in their original form can actually be interpreted as denoised core structures of the underlying graphs, and always refer to subgraphs of the original graphs. Therefore, to augment a given training set, the original definition of a matching-graph is not suitable. In [10], we propose an adapted version of a matching-graph that represents a mixed version of both original graphs, without being just a subgraph. However, this definition is still not optimal for the present purposes, since the resulting matching-graphs are always smaller than, or equal to, the original graphs. Hence, we now propose a further altered definition for matching-graphs more suited for the present context of increasing ensemble diversity.

The process of creating matching-graphs can be described as follows. Given a pair of graphs $g = (V, E, \mu, \nu)$ and $g' = (V', E', \mu', \nu')$, the *graph edit distance* is computed first[1]. The basic idea of graph edit distance is to transform g into g' using *edit operations* (*substitutions*, *deletions*, and *insertions* of both nodes and edges). We denote the substitution of two nodes $u \in V$ and $v \in V'$ by $(u \rightarrow v)$, the node deletion by $(u \rightarrow \varepsilon)$, and the node insertion by $(\varepsilon \rightarrow v)$, where ε refers to the empty node. By computing the graph edit distance one obtains a dissimilarity score $d(g, g')$, as well as a (sub-optimal) edit path $\lambda(g, g') = \{e_1, \ldots, e_s\}$ that consists of s edit operations that transform the source graph g in to the target graph g'.

Based on $\lambda(g, g')$ two matching-graphs $m_{g \times g'}$ and $m_{g' \times g}$ can now be built. Initially, $m_{g \times g'}$ and $m_{g' \times g}$ refer to the source graph g and the target graph g', respectively. In our procedure, we first define the partial edit path $\tau(g, g') = \{e_{(1)}, \ldots, e_{(t)}\} \subseteq \lambda(g, g')$ with $t = \lfloor p_1 \cdot s \rfloor$ edit operations, where $t < s$ is the amount of randomly selected edit operations from $\lambda(g, g')$ based on a certain probability $p_1 \in [0, 1]$[2]. Next, each edit operation $e_i \in \tau(g, g')$ is applied on graphs $m_{g \times g'}$ and $m_{g' \times g}$ according to the following three rules.

Case 1. e_i is a substitution $(u \rightarrow v)$: The labels of the matching nodes $u \in V$ and $v \in V'$ are exchanged in both $m_{g \times g'}$ and $m_{g' \times g}$. Note that this operation shows no effect, if the labels of the involved nodes are identical (i.e. $\mu(u) = \mu(v)$).

Case 2. e_i is a deletion $(u \rightarrow \varepsilon)$: Node $u \in V$

– is deleted in $m_{g \times g'}$.

[1] For this purpose, we use algorithm BP [14] with cubic time complexity.

[2] We use here the expression p_1 (with subscript 1), because later in the paper we will introduce a second probability p_2.

– is inserted in $m_{g' \times g}$.
  As we only execute parts of the edit path, it is possible that the adjacent nodes of u are not yet processed, which means that we do not know the edge structure of u in $m_{g' \times g}$. In this case, we perform a look-ahead to include edges from u to the corresponding nodes in $m_{g' \times g}$. Formally, for all node substitutions $(v \rightarrow u') \in \lambda(g, g')$, where node $v \in V$ is adjacent to node $u \in V$, we insert an edge between the inserted node u and node u' in $m_{g' \times g}$.

Case 3. e_i is an insertion $(\varepsilon \rightarrow v)$: Node $v \in V'$

– is deleted in $m_{g' \times g}$.
– is inserted in $m_{g \times g'}$ (using a similar look-ahead technique as defined for Case 2).

The basic rationale to apply these rules is that we aim at creating matching-graphs that are indeed related to the underlying graphs, but also substantially differ to them in significant ways. This is achieved by allowing both insertions of nodes and swappings of node labels.

Clearly, if p_1 is set to 1.0, $\tau(g, g')$ is equal to $\lambda(g, g')$, and thus all edit operations from the complete edit path are executed during the matching-graph creation. In this case, $m_{g' \times g}$ would be equal to the source graph g and $m_{g \times g'}$ would be equal to the target graph g'. For probabilities $p_1 < 1$, however, we obtain matching-graphs that are more diverse and particularly different from simple subgraphs (due to relabelled nodes and potential insertions). That is, when all edit operations of $\tau(g, g')$ are applied, both matching-graphs represent somehow intermediate graphs between g and g'.

Due to the flexibility of graph edit distance, the matching-graph can be built using graphs with any given labeling functions μ and ν. In other words it does not matter whether the graphs are unlabaled or contain categorical or continuous node and/or edge labels.

Figure 1 shows a visual example of an edit path $\lambda(g, g')$ between two graphs g and g' and two possible resulting matching-graphs $m_{g \times g'}$ and $m_{g' \times g}$. Both matching-graphs are created with the partial edit path that consists of $t = 3$ edit operations. In this example, it is clearly visible that neither $m_{g \times g'}$ nor $m_{g' \times g}$ is a subgraph of g or g', respectively. Furthermore, the effects of the look-ahead technique is visible. More specifically, between the inserted node $b \in V'$ and node $3 \in V$ an edge is inserted, even though the substitution $(3 \rightarrow c)$ is not yet carried out.

Note that the proposed process can lead to isolated nodes, despite look-ahead technique (for a detailed explanation of this phenomenon see [8]). As we aim to build graphs with nodes that are actually connected to at least one other node in the graph, we remove isolated nodes from the matching-graphs whenever they occur in our method.

2.2 Bagging with Matching-Graphs

Based on the process of creating matching-graphs for any pair of graphs, we can augment a given training set with virtually any number of additional graphs. In

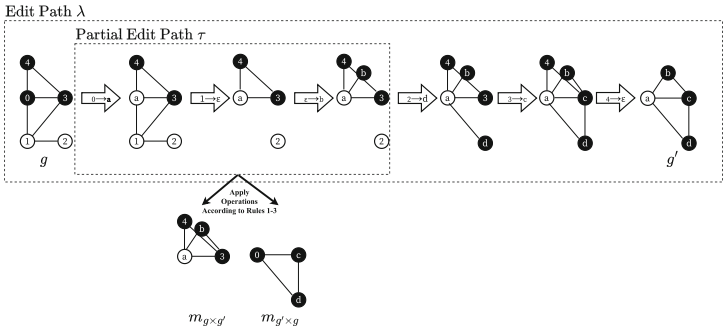

Fig. 1. An example of a complete edit path $\lambda(g, g')$, a partial edit path τ, and the resulting matching-graphs $m_{g\times g'}$ and $m_{g'\times g}$.

order to do this, we conduct the basic steps formalized in Algorithm 1 (which is similar in structure to the procedure described in [11]). The algorithm takes k sets of training graphs $G_{\omega_1}, \ldots, G_{\omega_k}$ stemming from k different classes $\omega_1, \ldots, \omega_k$, and builds two matching-graphs $m_{g\times g'}$ and $m_{g'\times g}$ for each possible graph pair $g, g' \in G_{\omega_i} \times G_{\omega_i}$. Note that the probability $p_1 \in [0.1, 0.9]$ used for the creation of the matching-graphs is randomly defined for each pair of graphs g, g' (see line 5). Assuming n training graphs per class ω_i this algorithm results in $n(n-1)$ matching-graphs, which are directly used to augment the corresponding training set G_{ω_i}. Hence, rather than n graphs, we now have access to $n(n-1)+n = n^2$ graphs per class ω_i[3].

Based on the augmented sets, a bagging ensemble $\mathcal{E} = \{c_1, \ldots, c_m\}$ with m classifiers can now be built. Each classifier $c_i \in \mathcal{E}$ is trained only on a subset of all training graphs. To this end, each classifier c_i of the ensemble $\mathcal{E}$ is trained on $\lfloor p_2 \times n^2 \rfloor$ randomly selected graphs from G_{ω_i}, where $p_2 \in [0, 1]$ is a predefined probability and n^2 is the number of graphs available in G_{ω_i} (i.e., we assume that G_{ω_i} is augmented to size $|G_{\omega_i}| = n^2$).

As base classifiers $c_i \in \mathcal{E}$, we use GNNs, which are – due to their inherent randomness – viable ensemble members. GNNs allow for the use of deep learning on graph structured data. The general goal of GNNs is to learn vector embeddings $h_v \in \mathbb{R}^n$ or $h_g \in \mathbb{R}^n$ that represent nodes $v \in V$ or complete graphs g, respectively. This vector space embedding can then be used for classification purposes. In order to learn an appropriate vector representation, GNNs implement a neighborhood aggregation strategy, called *neural message passing*, in which messages are exchanged between the nodes of a graph [15]. In the present paper, we employ a model that consists of *Graph Convolutional Layers* [16], denoted as

[3] By defining a further **for** loop inside the second **for** loop (in Algorithm 1 Line 5), just before the definition of p_1, even more than one matching-graph could be created for each pair of graphs, viz. we could produce more than $n(n-1)$ matching-graphs.

Algorithm 1: Graph Augmentation Algorithm

input : sets of graphs from k different classes $\mathcal{G} = \{G_{\omega_1}, \ldots, G_{\omega_k}\}$
output: same sets augmented by matching-graphs

foreach *set* $G_{\omega_i} \in \mathcal{G}$ **do**
  $M = \{\}$
  foreach *pair of graphs* $g, g' \in G_{\omega_i} \times G_{\omega_i}$ **do**
    Compute $\lambda(g, g') = \{e_1, \ldots, e_s\}$
    Randomly define p_1 in $[0.1, 0.9]$
    Define τ by selecting $\lfloor p_1 \cdot s \rfloor$ edit operations from λ
    Build both matching-graphs $m_{g \times g'}$ and $m_{g' \times g}$ according to τ
    $M = M \cup \{m_{g \times g'}, m_{g' \times g}\}$
  end
  $G_{\omega_i} = G_{\omega_i} \cup M$
end

GCN from now on[4]. For the final graph classification, we add a dropout layer and feed the graph embedding into a fully connected layer. The outputs of the individual classifiers are then aggregated into one single decision by means of majority voting.

3 Experimental Evaluation

3.1 Data Sets and Experimental Setup

The experimental evaluation is conducted on five data sets obtained from the TUDatasets repository[5]. The first three data sets contain graphs that represent chemical compounds (*NCI1*, *PTC(MR)* and *COX-2*). The fourth data set (*Cuneiform*) contains graphs that represent Hittie cuneiform signs[6]. The last data set (*Synthie*) is an algorithmically created data set. The graphs of the first three data sets consist of nodes labeled with discrete values and unlabelled edges, whereas both *Cuneiform* and *Synthie* contain real-valued continuous node labels and unlabeled edges. Each data set is split into a training and test set according to a 4:1 split.

The novel ensemble (denoted as GCN-e_{mg}) uses the augmented training data and is built as described in Sect. 2.2. We set p_2 to 0.3 and due to computational reasons we limit the amount of selected graphs to 100′000 per class. For each ensemble we create 100 classifiers, which are trained for 200 epochs (except for the *NCI1* data set, where we build 50 classifiers, trained for 50 epochs only, due to computational problems arising from the large number of graphs in this data set).

For all base classifiers (viz. GCNs) we use the Adam optimizer with an initial learning rate of 0.01, together with a CosineAnnealingLR scheduler. Furthermore, we use the Cross Entropy loss function. The batch size is set to 64. For

[4] Any other classifier could be used for the experiments, as long as both the reference ensemble and our novel ensemble are based on the same classifier.
[5] https://graphlearning.io.
[6] One of the oldest handwriting systems in the world.

the implementation of the ensemble we use the ensemble-pytorch library[7] which we adapted to seamlessly work with PyTorch Geometric [17][8].

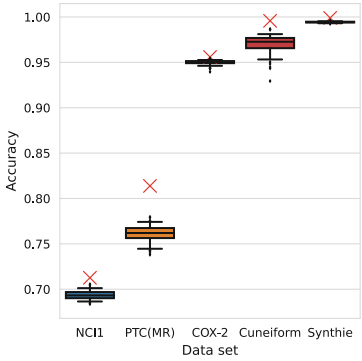

Fig. 2. Training accuracies of all individual classifiers of the ensemble for all data sets shown with a box-plot. The training accuracy of the ensemble is marked with a red cross. (Color figure online)

Figure 2 shows by means of box-plots the training accuracies of all individual classifiers available in the ensembles for all data sets. The diamonds above and below the boxes mark the 10% best and worst classifiers w.r.t. the accuracy. The training accuracy of the final ensemble is marked with a red cross. We observe that the diversity of the classifiers is the largest for NCI1, PTC(MR), and Cuneiform. It is also clearly visible that for all data sets the training accuracy of the complete ensemble is better than the accuracy of the best individual member. This is already a clear indication for the usefulness of the defined ensemble.

3.2 Reference Systems

The overall aim of the present experimental evaluation is to answer the question, whether or not matching-graphs can be beneficially employed to build robust classifier ensembles. In order to answer this research question, we use three reference systems for comparisons with our novel approach GCN-e_{mg}.

– *Reference system 1* (denoted by GCN): This reference system is trained on the full training set to obtain a baseline for the classification accuracy. In order to counteract uncontrolled randomness during initialization, each experiment that uses this reference system is repeated five times and the average accuracy is finally reported.

[7] https://ensemble-pytorch.readthedocs.io.
[8] https://pytorch-geometric.readthedocs.io.

We also perform an ablation study in order to empirically verify that the superiority of the proposed method is indeed based on the matching-graphs. To this end, we compare our novel ensemble GCN-e_{mg} with two additional reference systems.

- *Reference system 2* (denoted by GCN-e): This reference system is virtually the same as our novel ensemble but has only access to the original training data without matching-graphs.
- *Reference system 3* (denoted by SINGL-e_{mg}): This reference system refers to the best individual classifier of the novel augmented ensemble.

A comparison with reference system 2 allows us to better assess whether the matching-graphs, or the ensemble by itself, is the important element of the whole process. A comparison with reference system 3 allows us to assess whether the ensemble outperforms the randomly generated members of the system – in other words, whether the ensemble actually also makes a difference.

3.3 Test Results and Discussion

In Table 1 we compare the novel ensemble GCN-e_{mg} with the first reference system (a single GCN trained on the full training set). Remember that we run the GCN reference system five times to counteract randomness during initialization. This is why we report here the mean accuracies ($\pm$ standard deviation).

We observe that GCN-e_{mg} outperforms the reference system GCN in 18 out of 25 cases with statistical significance[9]. On the NCI1 data set, even though we observe an improvement in all five iterations, only four of them are statistically significant. On the PTC(MR) data set, we outperform the reference system in each iteration, however only two of the improvements are statistically significant. On the COX-2 data set we get an improvement in four out of five iterations (two of them are actually statistically significant). On Cuneiform and Synthie all of the improvements are statistically significant (10 out of 10 cases).

Table 1. Average classification accuracy of reference system 1 (GCN) compared to our novel ensemble (GCN-e_{mg}). Symbol ⓧ indicates a statistically significant improvement in x out of the five comparisons (using a Z-test at significance level $\alpha = 0.05$). Marked in bold is the best accuracy per data set.

	Ref. System 1	Ours
Dataset	GCN	GCN-e_{mg}
NCI1	70.5 ± 1.0	**74.0** ④
PTC(MR)	62.3 ± 4.5	**68.6** ②
COX-2	70.4 ± 11.1	**78.7** ②
Cuneiform	40.3 ± 20.5	**96.7** ⑤
Synthie	76.8 ± 7.5	**97.5** ⑤

[9] The statistical significance is computed via Z-test at significance level $\alpha = 0.05$.

Table 2. Classification accuracy of the ensemble without matching-graphs GCN-e, the individually best classifier SINGL-e$_{mg}$ and our ensemble with matching-graphs GCN-e$_{\text{mg}}$. Symbols ∘/∘ indicate a statistically significant improvement over the second/third reference system using a Z-test at significance level $\alpha = 0.05$). Marked in bold is the best accuracy per data set.

	Ref. System 2	Ref. System 3	Ours
Dataset	GCN-e	SINGL-e$_{mg}$	GCN-e$_{\text{mg}}$
NCI1	70.7	**74.8**	74.0 ∘/–
PTC(MR)	61.4	62.9	**68.6** ∘/∘
COX-2	73.4	77.6	**78.5** -/–
Cuneiform	68.3	93.3	**96.7** ∘/–
Synthie	64.2	93.8	**97.5** ∘/–

Next, in Table 2 we compare the novel ensemble with the other two reference systems (for the sake of an ablation study). First, we observe that the best single classifier of each ensemble (reference system 3) outperforms the baseline ensemble (reference system 2) on all data sets. This is a clear indication of the usefulness of the matching-graphs. Even more important, the proposed ensemble GCN-e$_{mg}$ outperforms the second reference ensemble GCN-e on all data sets. These improvements are statistically significant on four out of five data sets. This is a strong indication that the matching-graphs are the important factor in improving the classification accuracy, rather than primarily the ensemble itself. However, when comparing our ensemble with the third reference system, it is also obvious that the ensemble still makes an important contribution – only on NCI1 is the best individual classifier better than the ensemble.

4 Conclusion and Future Work

Ensemble learning is often able to improve the accuracy of single classification systems. One popular way to build an ensemble is bagging, which combines the output of many classifiers into one strong prediction. One of the main problems in building a robust ensemble is that large and diverse data sets are needed. In the present work we propose to use so-called matching-graphs to substantially increase the amount of training data available. On the basis of these augmented training sets of graphs, a classifier ensemble is built via bagging. As base classifiers for the ensemble we use GNNs (note, however, that any other classifier could be used as well).

By means of an experimental evaluation, we empirically confirm that our novel approach significantly outperforms three related reference systems, viz. a single GNN classifier, a bagging ensemble trained on the original training set, as well as the best individual classifier stemming from the novel ensemble. Hence, we conclude that matching-graphs provide a versatile way to generate large sets of additional graphs in order to build a diverse and robust ensemble.

For future work we see at least two rewarding avenues that can be pursued. First, we could explore other ensemble modalities (rather than bagging), and second, other aggregation techniques to combine the results could be explored (rather than majority voting).

References

1. Foggia, P., Percannella, G., Vento, M.: Graph matching and learning in pattern recognition in the last 10 years. Int. J. Pattern Recogn. Artif. Intell. **28**(1), 1450001 (2014). https://doi.org/10.1142/S0218001414500013
2. Conte, D., Foggia, P., Sansone, C., Vento, M.: Thirty years of graph matching in pattern recognition. Int. J. Pattern Recogn. Artif. Intell. **18**(3), 265–298 (2004). https://doi.org/10.1142/S0218001404003228
3. Gao, X., Xiao, B., Tao, D., Li, X.: A survey of graph edit distance. Pattern Anal. Appl. **13**(1), 113–129 (2010). https://doi.org/10.1007/s10044-008-0141-y
4. Kang, U., Hebert, M., Park, S.: Fast and scalable approximate spectral graph matching for correspondence problems. Inf. Sci. **220**, 306–318 (2013). https://doi.org/10.1016/j.ins.2012.07.008
5. Kriege, N.M., Johansson, F.D., Morris, C.: A survey on graph kernels. Appl. Netw. Sci. **5**(1), 6 (2020). https://doi.org/10.1007/s41109-019-0195-3
6. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Trans. Neural Netw. **20**(1), 61–80 (2009). https://doi.org/10.1109/TNN.2008.2005605
7. Dong, X., Yu, Z., Cao, W., Shi, Y., Ma, Q.: A survey on ensemble learning. Front. Comput. Sci. **14**(2), 241–258 (2020). https://doi.org/10.1007/s11704-019-8208-z
8. Fuchs, M., Riesen, K.: A novel way to formalize stable graph cores by using matching-graphs. Pattern Recogn. **131**, 108846 (2022). https://doi.org/10.1016/j.patcog.2022.108846
9. Fuchs, M., Riesen, K.: Iterative creation of matching-graphs - finding relevant substructures in graph sets. In: Proceedings of the 25th Iberoamerican Congress on Pattern Recognition, CIARP25 2021 (2021)
10. Fuchs, M., Riesen, K.: Graph augmentation for small training sets using matching-graphs. In: ICPRAI - 3rd International Conference on pattern Recognition and Artificial Intelligence (2022)
11. Fuchs, M., Riesen, K.: Graph augmentation for neural networks using matching-graphs. In: Gayar, N.E., Trentin, E., Ravanelli, M., Abbas, H. (eds.) Artificial Neural Networks in Pattern Recognition - 10th IAPR TC3 Workshop, ANNPR 2022, Proceedings. Lecture Notes in Computer Science, Dubai, United Arab Emirates, 24–26 November 2022, vol. 13739, pp. 3–15. Springer, Heidelberg (2022). https://doi.org/10.1007/978-3-031-20650-4_1
12. Hastie, T., Rosset, S., Zhu, J., Zou, H.: Multi-class adaboost. Stat. Interface **2**(3), 349–360 (2009)
13. Breiman, L.: Bagging predictors. Mach. Learn. **24**(2), 123–140 (1996). https://doi.org/10.1007/BF00058655
14. Riesen, K., Bunke, H.: Approximate graph edit distance computation by means of bipartite graph matching. Image Vis. Comput. **27**(7), 950–959 (2009). https://doi.org/10.1016/j.imavis.2008.04.004
15. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: International Conference on Machine Learning, pp. 1263–1272. PMLR (2017)

16. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: 5th International Conference on Learning Representations, ICLR 2017, Conference Track Proceedings, Toulon, France, 24–26 April 2017. OpenReview.net (2017). https://openreview.net/forum?id=SJU4ayYgl
17. Fey, M., Lenssen, J.E.: Fast graph representation learning with PyTorch Geometric. In: ICLR Workshop on Representation Learning on Graphs and Manifolds (2019)