Relation order histograms as a network embedding tool*

Radosław Łazarz[1][0000−0002−3151−9759] and Michał Idzik[1][0000−0002−8446−4966]

Institute of Computer Science, AGH University of Science and Technology, Kraków, Poland
{lazarz,miidzik}@agh.edu.pl

Abstract. In this work, we introduce a novel graph embedding technique called NERO (Network Embedding based on Relation Order histograms). Its performance is assessed using a number of well-known classification problems and a newly introduced benchmark dealing with detailed laminae venation networks. The proposed algorithm achieves results surpassing those attained by other kernel-type methods and comparable with many state-of-the-art GNNs while requiring no GPU support and being able to handle relatively large input data. It is also demonstrated that the produced representation can be easily paired with existing model interpretation techniques to provide an overview of the individual edge and vertex influence on the investigated process.

Keywords: Graph classification · Graph embedding · Representation learning · Complex networks.

1 Introduction

Due to their expressiveness and flexibility, complex networks appear in numerous real-world applications, such as sociology, bibliometrics, biochemistry, or telecommunications (to name a few) [36]. However, there are practical issues (i.a. lack of a universal notion of similarity, the fact that there are no bounds on the total vertex number or the number of single vertex neighbours) that make utilisation of such data representation challenging and cumbersome, especially in the context of machine learning. Thus, finding a way to bridge the gap between the graph domain and the vector-driven mainstream science is one of the most important topics in structural pattern processing [4].

1.1 Related solutions

Classical answers to that problem tend to utilise the concept of graph kernels — either explicit (bivariate functions whose results are equivalent to graph inner

* The research presented in this paper was financed using the funds assigned by the Polish Ministry of Science and Higher Education to AGH University of Science and Technology.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_18

products) or implicit (operators mapping the graphs into the vector spaces, where there is a wide range of kernel functions available). Notable examples include counting the number of matching shortest path in input graphs [3], comparing estimated distributions of graphlets (small subgraphs with up to 5 nodes) [28], or relying on pairwise Wasserstein distances between graphs in the dataset [30].

Recently, there was a resurgence of interest in graph representation learning. Deep convolutional neural networks achieved tremendous results in computer vision tasks and, as a consequence, numerous works tried to adapt the ideas behind their success to the graph analysis domain [36]. While differing in implementation details, those approaches are collectively referred to as graph neural networks (GNNs). Multiple directions are being currently pursued in the field, such as imitating the popular U-Net architectures [12], leveraging the transformer self-attention mechanism [22], designing adaptive pooling operators [35], employing Gaussian mixture models [16], or utilising anchor graphs [1].

Nonetheless, despite the considerable achievements of GNN techniques, one should remember that their effectiveness comes with a high demand for memory, computational power, and specialised hardware. Hence, it may still be of interest to seek progress with kernel methods, especially in case of large networks.

2 Basic notations

Graph $G = (\mathbf{V}, \mathbf{E}, \mathbf{T_V}, \mathbf{T_E})$ is an ordered tuple comprised of a finite ordered set of vertices $\mathbf{V} = \{v_1, \ldots, v_{n_V}\}$, a finite ordered set of edges $\mathbf{E} = \{e_1, \ldots, e_{n_E}\} \mid \mathbf{E} \subseteq \mathbf{V} \times \mathbf{V}$, a finite ordered set of vertex trait functions $\mathbf{T_V} = \{t_{V_1}, \ldots, t_{V_{n_{T_V}}}\}$, and a set of edge trait functions $\mathbf{T_E} = \{t_{E_1}, \ldots, t_{E_{n_{T_E}}}\}$, such that $\forall_{t_V \in \mathbf{T_V}}\ \mathrm{dom}(t_V) = \mathbf{V}$ and $\forall_{t_E \in \mathbf{T_E}}\ \mathrm{dom}(t_E) = \mathbf{E}$. We call given trait a label if its codomain is discrete, or an attribute if it is continuous. Throughout this work we assume that all graphs are undirected, i.e. $(v, u) \in \mathbf{E} \implies (u, v) \in \mathbf{E}$.

A k-dimensional array of size m_1-by-$\cdots$-by-m_k is denoted as $\mathbf{A} = S^{m_1 \times \cdots \times m_k}$, where S is the set of possible element values. $\mathbf{A}_{i_1,\ldots,i_k}$ represents its element on position $i_1, \ldots, i_k$. Using $*$ as a position component represents taking all elements along the given dimension — e.g. $\mathbf{A}_{*,j}$ is the j^{th} column of $\mathbf{A}$. The notation $(\mathbf{A}_{i_1,\ldots,i_k})_C$, where C is the set of conditions, is used to describe an array sub-fragment — e.g. $(\mathbf{A}_{i,j})_{1 \le i \le 3,\, 1 \le j \le 3}$ is the upper left 3-by-3 sub-array. Additionally, $\mathbf{0}^{m_1 \times \cdots \times m_k}$ is the m_1-by-$\cdots$-by-m_k array filled with zeros.

3 Relation order embedding

The main idea behind the proposed embedding method is inspired by previous research by Bagrow et al. [2] and the later improvements suggested by Czech et al [6–8]. The former introduced the concept of a B-matrix — a way of portraying an arbitrary network that contains information about its structure at multiple

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_18

levels of abstraction and allows to express between-graph similarity as the euclidean distance. For a given graph, the B-matrix $B_{k,l}$ contains the number of vertices that have a degree of l in the distance-k-graph corresponding to the k^{th} matrix row. A distance-k-graph, in turn, contains all the original nodes and connects with edges those of them that were initially separated by the shortest path distance equal to k. The latter expanded the framework and demonstrated that it is possible to improve its performance by utilising the node-edge distance [6] or employing alternative centrality measures [7]. It also explored the possibilities of obtaining embeddings of large graphs in a distributed fashion [8]. However, all those attempts were focused on purely structural data and assumed the lack of labels or attributes, severely limiting their applicability.

This work aims to further develop the aforementioned techniques and adapt them to support graphs with any type of traits. It is based on the observation, that two crucial components are common for all of them: assembling the embeddings from feature histograms (thus making them invariant to changes in edge and node processing order) and doing so not only for a starting graph but for the whole distance-k family (therefore capturing both local and global relations between elements).

We start by reducing the problem to a vertex-traits-only one by converting the input graph to an edge-node bipartite form (see Algorithm 3.1). This decision not only simplifies the subsequent steps but also potentially boosts the expressive power of the method (as shown in [6]).

Algorithm 3.1 BipartiteEdgeVertexGraph(G)

$(\mathbf{V}, \mathbf{E}, \mathbf{T_V}, \mathbf{T_E}) \leftarrow G$
$\mathbf{V_B} = \mathbf{V} \cup \mathbf{E}$
$\mathbf{E_B} = \{(v, e) \mid v \in \mathbf{V} \wedge e \in \mathbf{E} \wedge v \in e\}$
$$\mathbf{T_B} = \left\{ \left(\mathbf{V_B} \ni v \mapsto t_B(v) = \begin{cases} t(v) & v \in \mathrm{dom}(t) \\ \varnothing & \text{otherwise} \end{cases} \right) \middle| \; t \in \mathbf{T_V} \cup \mathbf{T_E} \right\}$$
$G_B \leftarrow (\mathbf{V_B}, \mathbf{E_B}, \mathbf{T_B}, \varnothing)$
return G_B

Next, we need to introduce the notion of elements relation order. Two graph elements are in a relation of order $r+1$ with one another if they are separated by a distance r in the corresponding graph. As a consequence, a node is in order-1 relation with itself, order-2 relation with edges incident to it, and order-3 relation with its direct neighbours.

Now, it is possible to define in the broadest terms the excepted form of the discussed embedding output. NERO (Network Embedding based on Relation Order histograms) should produce a three-dimensional tensor $\mathbf{N}$, where $\mathbf{N}_{i,j,k}$ is the number of order-i relations between vertices for which the first relation element is annotated with trait value j and the second with trait value k. If there is only a single discrete trait present, then $\mathbf{N}_{3,*,*}$ is equivalent to a standard graph attribute matrix, and $\mathbf{N}_{2k+1,*,*}$ to its distance-k graph incarnations.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_18

As we can see, the core architecture of the previously discussed techniques is still upheld. There is a histogram as a basic building block (but this time of trait connectivity, not of node centrality) and focus enumerating and representing multiple levels of abstraction (generalised as a relation order). However, such general definition necessarily omits important practical details (e.g. sometimes traits are continuous and long-range relations often don't need the same representational resolution as the close ones) — the subsequent sections provide a more comprehensive overview of the embedding algorithm and its intended implementation.

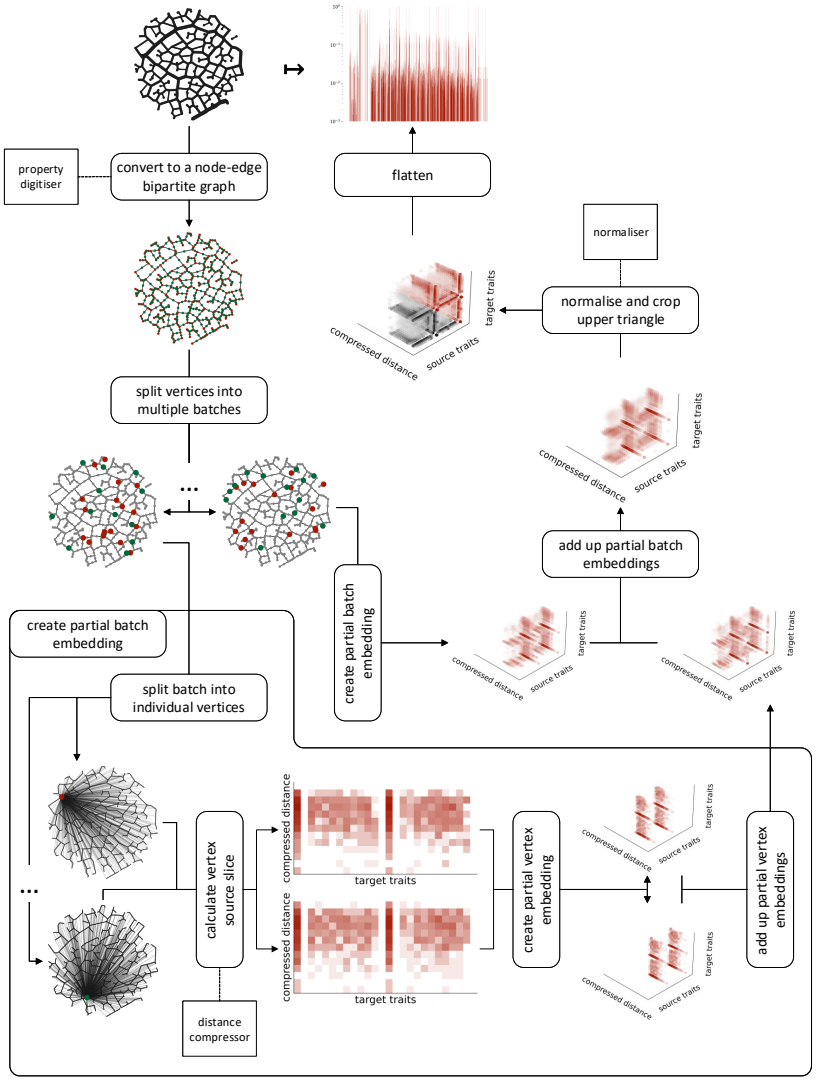


Fig. 1: NERO graph embedding workflow.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_18

3.1 NERO embedding scheme

The general flow of the proposed embedding scheme is outlined as Algorithm 3.2 and visualised in Figure 1. The first step is the already discussed conversion to a bipartite graph form G'. Next, it is necessary to digitise any traits annotating the graph vertices — a procedure resulting in the matrix $\mathbf{T}'$, where $\mathbf{T}'_{i,j}$ is the single-integer representation of the trait T_j value for vertex v_i. The actual digitisation technique is a modular part of the solution and can be chosen with a particular problem in mind (some of them might require prior supervised or unsupervised training). The specific variants utilised in this work are described in Section 3.2. One mandatory condition that has to be satisfied by any applied digitiser is that there should be no overlaps between numbers assigned to values among different traits ($\forall_{j_1,j_2}\ j_1 < j_2 \implies \max \mathbf{T}'_{*,j_1} < \min \mathbf{T}'_{*,j_2}$), so they are not mixed in the final outcome. We also store the value $t_{\max} = \max \mathbf{T}'$ for later usage.

Algorithm 3.2 EmbedGraph(G, n_b)

$(\mathbf{V}, \mathbf{E}, \mathbf{T}, \varnothing) \leftarrow G' \leftarrow$ BipartiteEdgeVertexGraph(G)
$(\mathbf{T}', t_{\max}) \leftarrow$ DigitisedTraits$(\mathbf{T})$
$\delta_{\max} = 1$
$\mathbf{N} \leftarrow \mathbf{0}^{\delta_{\max} \times t_{\max} \times t_{\max}}$
for $\mathbf{V^b} \in$ VertexBatches$(\mathbf{V}, n_b)$ **do**
  $\mathbf{N^b} \leftarrow \mathbf{0}^{\delta_{\max} \times t_{\max} \times t_{\max}}$
  for $v_i \in \mathbf{V^b}$ **do**
    $(\mathbf{N^{b,v}}, d_{\max}) \leftarrow$ VertexSourceSlice$(v, \mathbf{V}, \mathbf{E}, \mathbf{T}', t_{\max})$
    if $d_{\max} > \delta_{\max}$ **then**
      $\delta_{\max} = d_{\max}$
      $\mathbf{N^b} \leftarrow$ EnlargeEmbedding$(\mathbf{N^b}, \delta_{\max})$
    for $j \in \mathbf{T}'_{i,*}$ **do**
      $\left(\mathbf{N^b}_{d,j,*}\right)_{1 \le d \le d_{\max}} \leftarrow \left(\mathbf{N^b}_{d,j,*}\right)_{1 \le d \le d_{\max}} + \mathbf{N^{b,v}}$
  $\mathbf{N} \leftarrow$ EnlargeEmbedding$(\mathbf{N}, \delta_{\max})$
  $\mathbf{N} \leftarrow \mathbf{N} + \mathbf{N^b}$
return $\mathbf{N}$

After that, the result $\mathbf{N}$ is initialised with zeros and the main phase of the procedure begins. Vertices are divided into n_b batches of chosen size to enable the possibility of map-reduce style parallel computations (every batch can be processed independently). Then, for each vertex v in a batch, a source slice is calculated, as specified in Algorithm 3.3.

The goal of the slice computation is to create matrix $\mathbf{N^{b,v}}$ containing information about all the relations in which the current vertex is the first element ($\mathbf{N^{b,v}}_{i,j}$ is the number of vertices with trait value j in order-i relation with the vertex v). To do so, we first need to calculate all the pairwise distances between the current vertex and all the other ones (e.g. by BFS graph traversal). However, using raw distance values turns out to be impractical unless the graph has

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_18

Algorithm 3.3 VertexSourceSlice$(v, \mathbf{V}, \mathbf{E}, \mathbf{T}', t_{\max})$

$\mathbf{D} \leftarrow$ ShortestDistances$(v, \mathbf{V}, \mathbf{E})$
$(\mathbf{D}', d_{\max}) \leftarrow$ CompressedDistances$(\mathbf{D})$
$\mathbf{N^{b,v}} \leftarrow \mathbf{0}^{d_{\max} \times t_{\max}}$
for $i \in \{1, 2, \ldots, n_V\}, j \in \{1, 2, \ldots, n_T\}$ **do**
  $\mathbf{N^{b,v}}_{\mathbf{D'}_i, \mathbf{T'}_{i,j}} \leftarrow \mathbf{N^{b,v}}_{\mathbf{D'}_i, \mathbf{T'}_{i,j}} + 1$
return $(\mathbf{N^{b,v}}, d_{\max})$

a low diameter. Instead, the distances are compressed using one of the options presented in Section 3.3.

The slices are then added to appropriate regions of tensor $\mathbf{N^b}$, iteratively creating a partial embedding. After that, the batch outcomes are summed up position-wise and form a complete result $\mathbf{N}$ (the accumulator is resized whenever necessary based on current $\delta_{\max}$. Finally, the obtained values are normalised (as described in Section 3.4) and reshaped into a flat vector to facilitate later usage.

3.2 Trait digitisation

Discrete traits were digitised in the simplest possible way — by consecutively assigning a natural number to a value whenever a new one is encountered. In the case of continuous traits, two alternatives sharing the same basic scheme were tested. They both utilise a fixed number n_h of non-overlapping bins with boundaries determined by fitting them to the training data. Trait values are then mapped to the number of the bin they fit in, with two special numbers reserved for those belonging outside of the bin scope and for vertices without specified trait value.

The first variant (annotated $\varnothing$ and commonly known as equal-width bins [14]) was used as a baseline. It finds the minimal and maximal trait value in the set and then divides that range into same-size chunks. The second one (annotated A) tries to adapt bin edges to the way the values are distributed. It starts by estimating the said distribution — to do so it uses $n_i \gg n_h$ equal-width bins to create a value histogram for each sample and then calculates their mean over the whole training set. Finally, it picks edge positions so that there would be an equal probability of falling into each one of them.

3.3 Relation order compression

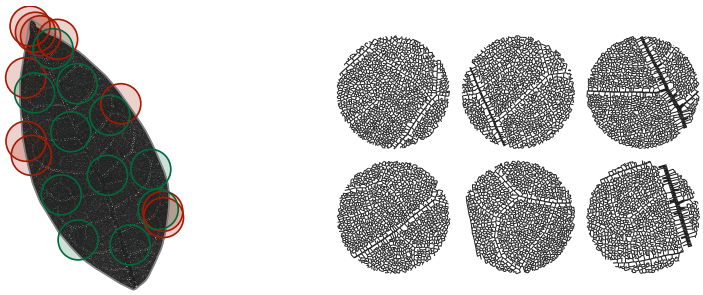
Relation order compression is a non-linear mapping between the actual between-vertex distance and the declared relation order. The rationale for employing it is twofold. Firstly — it directly reduces the dimensionality of the resulting embedding, decreasing the memory requirements and speeding up the later operations. Secondly — the experiences from the computer vision field [29] strongly suggest that the higher the relation order, the less important it is to know its exact value (i.e. if a vertex is on the opposite side of the graph it might not be crucial to differentiate whether it lies exactly one step further than another one or not).

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_18

We utilised three types of compression: no compression at all ($\varnothing$) as a baseline, and two bin-based compression schemes similar to trait digitising ones discussed in the previous section. Bin edges b_i were determined by the following formulas: $b_i = \textsc{Fibonacci}(\mathrm{i})$ (a standard Fibonacci sequence, denoted by F) and $b_{i-1} = \lfloor a^i i + bi + c \rfloor$ with $a = 1.074$, $b = 1.257$, $c = -3.178$ (denoted by V, a solution inspired by VGG19 [29] CNN layers receptive field sizes). Whichever one was used, the first two boundaries were set to fixed values of $b_0 = 0$ and $b_1 = 1$, while target number of bins was set to a fixed value n_r.

3.4 Normalisation

The relative values of individual NERO embedding components differ in magnitude, as the relation order density is not uniform (there are much more moderately distanced vertices than those in close vicinity or in highly eccentric pairs). Thus, it is practical to normalise them before any further processing. Throughout this work, all values were always subject to the so-called standard scaling — i.e. standardised by removing the mean and scaling to unit variance (baseline $\varnothing$). Additionally, they were sometimes scaled relative to the maximum value per given relation order — $\forall_i \, \mathbf{N}_{i,**} \leftarrow \frac{\mathbf{N}_{i,**}}{\max \mathbf{N}_{i,**}}$ (variant R).



(a) Accepted and discarded circles. (b) Examples of cropped fragments.

Fig. 2: Obtaining leaf fragment samples.

4 Laminae venation graphs

Leaf venation networks of flowering plants form intricate anastomosing structures, that evolved to serve numerous functions, from efficient water transportation to mechanical reinforcement of the lamina [26]. As a result, they are intensively studied by numerous research fields, such as plant paleoecology, plant physiology, developmental biology, or species taxonomy [24].

From the graph embedding point of view, processing of such networks creates an interesting computational challenge - not only because of the hefty numbers of vertices and edges but also due to their large radii making shortest path

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_18

computations significantly harder [18]. However, popular datasets tend to focus almost solely on data coming from either biochemistry, computer vision, or social sciences [20]. In this work, we introduce an additional set of novel benchmark problems that require correct recognition of organism genus on the basis of small vasculature fragments.

The results obtained by Ronellenfitsch et al. [25] were utilised as the foundation for the subsequent sample generation. They consist of 212 venation graphs belonging to 144 species from 37 genera and were obtained by a multi-stage procedure, involving chemical staining, high-resolution scanning and numerous image transformations. Unfortunately, the samples are strongly unbalanced, with 102 belonging to *Protium* and 25 to *Bursera* genus. Furthermore, it was necessary to remove 44 of them due to various defects caused by unsuccessful extraction or accidental laminae damage. The remaining ones are highly reticulate large planar graphs comprised of up to 300000 elements, with two continuous edge attributes — their lengths and diameters.

Algorithm 4.1 CropSamples$(G, n_a, \tau_{A_c}, \tau_{\varrho_v}, \tau_{r_c})$

$S_c \leftarrow \varnothing$
$(\mathbf{V}, \mathbf{E}, \{T_x\}, \mathbf{T_E}) \leftarrow G$
$\mathbf{V^u} \leftarrow \mathbf{V}$
$\mathbf{X} \leftarrow \{T_x(v) \mid v \in \mathbf{V}\}$
$\mathbf{H} \leftarrow$ ConvexHull$(\mathbf{X})$
$A \leftarrow$ ShoelaceArea$(\mathbf{H})$
$\varrho_v \leftarrow \frac{|\mathbf{V}|}{A}$
$A_c \leftarrow \tau_{A_c} A$
$r_c \leftarrow \sqrt{\frac{A_c}{\pi}}$
while $n_a > 0 \ \wedge \ |\mathbf{V^u}| > 0$ **do**
  $v_c \leftarrow$ RandomChoice$(\mathbf{V^u})$
  $x_c \leftarrow T_x(v_c)$
  $\mathbf{V^c} \leftarrow \{v \mid v \in \mathbf{V} \ \wedge \ \|T_x(v) - x_c\| < r_c\}$
  if $\frac{|\mathbf{V^c}|}{A_c} > \tau_{\varrho_v} \varrho_v$ **then**
    $\mathbf{V^u} \leftarrow \mathbf{V^u} \setminus \{v \mid v \in V \ \wedge \ \|T_x(v) - x_c\| < \tau_{r_c} r_c\}$
    $\mathbf{E^c} \leftarrow \{e = (v_b, v_e) \mid e \in E \ \wedge \ \{v_b, v_e\} \subseteq V_c\}$
    $G_c \leftarrow (\mathbf{V^c}, \mathbf{E^c}, \{T_x\}, \mathbf{T_E})$
    $S_c \leftarrow S \cup \{G_c\}$
  else
    $\mathbf{V^u} \leftarrow \mathbf{V^u} \setminus \{v_c\}$
    $n_a \leftarrow n_a - 1$
return S_c

Algorithm 4.1 was then employed in order to convert those networks into the final datasets by cropping out meaningful circular fragments. It starts by finding the convex hull $\mathbf{H}$ of a given venation graph. Next, the area A of the obtained polygon is calculated using the well-known shoelace formula and used

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_18

to estimate the vertex density ϱ_v together with the target cropping radius r_c (the latter depending on the target area percentage parameter τ_{A_c}). Then, a central vertex v_c is randomly selected from the available vertices pool $\mathbf{V^u}$ and together with all its neighbours up to the distance r_c forms a crop candidate set $\mathbf{V^c}$.

If the candidate density is higher than a chosen percentage τ_{ϱ_v} of the average one, the crop area is considered to be sufficiently filled with content. In this case, the vertex set is supplemented by all the edges $\mathbf{E^c}$ incident to its members and forms a new sample G_c. After that, vertices closer than τ_{r_c} times the radius r_c from the centre are excluded from the $\mathbf{V^u}$ pool to ensure the lack of overlaps between the produced samples. On the other hand, if the density criterion is not satisfied only vertex v_c leaves the pool and the attempt is counted to be a failed one. The aforementioned loop is repeated until there are n_a failed attempts registered and then proceeds to the next leaf. Figure 2a depicts a possible outcome of the procedure (green circles are accepted and the red ones are omitted due to containing excessive free space) and Figure 2b shows examples of isolated fragments.

Presented cropping scheme enables the production of venation datasets with chosen fragments radii and distribution, three of which were used to evaluate the proposed embedding technique. Each of them was generated with a different focus in mind: LaminaeBig ($\tau_{A_c} = 0.05$, $\tau_{\varrho_v} = 0.7$, $\tau_{r_c} = 2.0$) contains a smaller number of densely packed graphs with large diameter, LaminaeSmall ($\tau_{A_c} = 0.005$, $\tau_{\varrho_v} = 0.7$, $\tau_{r_c} = 2.0$) increases samples count while decreasing their size, and LaminaeLight ($\tau_{A_c} = 0.01$, $\tau_{\varrho_v} = 0.9$, $\tau_{r_c} = 3.0$) is intended to have lower computational demands due to a moderate number of high quality smaller radius samples. A brief summary of their characteristics can be found in Table 1. The cropping attempts threshold was always set to $n_a = 100$.

5 Experiments and validation

Multiple configurations of the proposed algorithm are utilised throughout this section. Whenever it happens, the specific version is stated using the $\text{NERO}_{D,C,N}$ signature, where D is the chosen trait digitisation method, C specifies the relation order compression scheme, and N is the utilised normalisation. All calculations were performed using a Ryzen 7 3700X 3.6GHz CPU. GNNs training was supported by a GeForce RTX 2080Ti 11GB GPU.

5.1 Graph classification

The graph classification task is one of the standard ways of assessing representation robustness. In order to evaluate our proposed embedding scheme, we performed a series of tests on a diverse selection of benchmark problems, including the popular options from TUDataset [20] and three new ones introduced in this work. The first group contains macromolecular graphs representing the spatial arrangement of amino acid structures (Proteins [11], DD [27]) or small molecules described as individual atoms and chemical bonds between

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_18

Table 1: Benchmark datasets characteristics.

Name	Graphs	Classes	Average Nodes	Average Edges	Node Labels	Edge Labels	Node Attrib.	Edge Attrib.
DD	1178	2	248.32	715.66	+ (1)	-	-	-
Mutag	188	2	17.93	19.79	+ (1)	+ (1)	-	-
Proteins	1113	2	39.06	72.82	+ (1)	-	+ (1)	-
NCI1	4110	2	29.87	32.30	+ (1)	-	-	-
LaminaeBig	2044	36	3585.35	4467.58	-	-	-	+ (2)
LaminaeSmall	17924	36	392.67	473.82	-	-	-	+ (2)
LaminaeLight	4110	36	820.74	1004.31	-	-	-	+ (2)

them (Mutag [9], NCI1 [32]) and the second focuses on large-diameter laminae venation networks (Section 4 presents a more detailed overview of its origins). A brief comparison of their key characteristics is collected in Table 1.

The testing procedure was arranged following the guidelines presented in [20]. Accuracy scores were obtained as a result of stratified 10-fold cross-validation. For each split the training set was further subdivided into the actual training set and an additional validation set, the latter consisting of 10% of the initial samples and keeping the class frequencies as close as possible to the original ones. The optimal parameters of each utilised algorithm were estimated using a grid search over sets of available values with validation accuracy used as a ranking basis. The winning configuration was then retrained once more on the larger group of training samples and evaluated on the test batch previously set aside. Finally, the result was calculated as a mean of the 10 partial ones acquired in an aforementioned way.

In case of the classic benchmarks (Table 2), there were 3 repetitions of the said experiment — the reported value is the average outcome of the whole cross-validation procedure and the $\pm$ notation is employed to show the standard deviation of the mean accuracy. Obtained scores are compared with those achieved by a wide range of well-established solutions, representing both the graph kernel family (WL [27], GR [28], SP [3], WL-OA [17], WWL [30]) and the GNN approach (GIN-ϵ [33], GIN-ϵ-JK [34], U2GNN [22], g-U-Net [12], DDGK [1], GIC [16], PPGN [19], HGP-SL [35], GraphSAGE [15], GAT [31], EdgePool [10], PSCN [23]). The sources of the provided values are referenced in corresponding rows. Due to differences in applied measurement methodologies, the declared errors should not be directly compared with one another and should instead be treated only as an approximate credibility gauge. The scores are declared as not available (N/A) if they were not reported in any publications known to the authors (in case of algorithms our solution is being compared to)[1], would make no sense in the context (different trait digitisation techniques when there are no continuous attributes present)[2], or were not computationally feasible (e.g. no relation order compression for large graphs)[3].

NERO runs used an Extremely Randomised Trees [13] classifier with number of decision sub-trees equal to $n_X = 5000$ as its final step. Number of digitiser

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_18

Table 2: Classification accuracies on popular TUDatasets benchmarks.

	DD	MUTAG	PROTEINS	NCI1
WL[30]	78.29%±0.30%	85.78%±0.83%	74.99%±0.28%	85.83%±0.09%
GR[22, 20]	78.45%±0.26%	81.58%±2.11%	71.67%±0.55%	66.0%±0.4%
SP[35, 21, 20]	78.72%±3.89%	85.8%±0.2%	75.6%±0.7%	74.4%±0.1%
WL-OA[30]	79.15%±0.33%	87.15%±1.82%	**76.37%±0.30%**	**86.08%±0.27%**
WWL[30]	**79.69%±0.50%**	**87.27%±1.50%**	74.28%±0.56%	85.75%±0.25%
GIN-ϵ[20]	71.90%±0.70%	84.36%±1.01%	72.2%±0.6%	77.7%±0.8%
GIN-ϵ-JK[20]	73.82%±0.97%	83.36%±1.81%	72.2%±0.7%	78.3%±0.3%
U2GNN[22]	80.23%±1.48%	89.97%±3.65%	78.53%±4.07%	N/A[1]
g-U-Net[12]	82.43%	N/A[1]	77.68%	N/A[1]
DDGK[1]	**83.1%±12.7%**	91.58%±6.74%	N/A[1]	68.10%±2.30%
GIC[16]	N/A[1]	**94.44%±4.30%**	77,65%±3.21%	**84.08%±1.77%**
PPGN[19]	N/A[1]	90.55%±8.70%	77,20%±4,73%	83.19%±1.11%
HGP-SL[35]	80.96%±1.26%	N/A[1]	**84.91%±1.62%**	78.45%±0.77%
SAGE[35, 22]	75.78%±3.91%	79.80%±13,9%	74.01%±4,27%	74.73%±1.34%
GAT[35, 22]	77.30%±3.68%	89.40%±6,10%	74.72%±4,01%	74.90%±1.72%
EdgePool[35]	79.20%±2.61%	N/A[1]	82.38%±0.82%	76.56%±1.01%
PSCN[16]	N/A[1]	92.63%±4.21%	75.89%±2.76%	78.59%±1.89%
NERO$_{A,V,R}$	79.40%±0.49%	86.49%±0.89%	**77.89%±0.60%**	80.40%±0.25%
NERO$_{A,F,R}$	**80.45%±0.18%**	86.55%±0.25%	77.30%±0.30%	80.01%±0.44%
NERO$_{A,V,\varnothing}$	79.15%±0.28%	**88.68%±0.25%**	76.82%±0.44%	**81.63%±0.14%**
NERO$_{A,F,\varnothing}$	79.40%±0.26%	88.10%±1.32%	76.88%±0.19%	80.89%±0.29%
NERO$_{A,\varnothing,R}$	N/A[3]	85.64%±0.41%	76.55%±0.19%	79.80%±0.20%
NERO$_{\varnothing,V,R}$	N/A[2]	N/A[2]	75.68%±0.29%	N/A[2]

bins was chosen from $n_h \in \{5, 10, 20\}$ (with $n_i = 500$ where applicable), similarly for the relation order compression $n_r \in \{5, 10, 20\}$.

For the laminae benchmarks (Table 3) only a single experiment run was performed and the ± symbol denotes the standard deviation of the accuracy itself (estimated over the 10 cross-validation splits). As a consequence, the potential error of such estimation is much higher that the one demonstrated for the previous benchmark group. The NERO parameters were set to $n_h = 10$ and $n_r = 20$ and the RBF kernel SVM [5] with L2 penalty parameter $C \in \{0.1, 1.0, 10.0, 100.0\}$ was the classifier of choice. Some scores were declared as not available (N/A) due to exceeding per-algorithm computation time limit (12 hours)[4]. Methods supporting only discrete edge labels had them provided after being digitised by scheme A.

5.2 Results interpretability

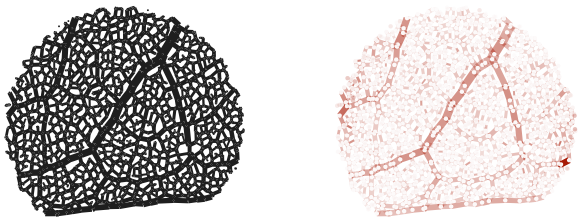
An additional feature of the proposed NERO embedding is the fact, that the final tensor $\mathbf{N}$ is always the combination of individual node and edge contributions $\mathbf{N}^{\mathbf{b},\mathbf{v}}$. This, in turn, enables few useful routines. First, as already mentioned, it is possible to parallelise the computation in accordance with the map-reduce

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_18

Table 3: Classification accuracies on laminae fragments benchmark datasets.

	LaminaeBig	LaminaeSmall	LaminaeLight
WL	71.6%±2.0%	**66.9%±1.5%**	66.0%±2.7%
GR	**73.5%±2.7%**	65.9%±1.5%	64.7%±3.4%
SP	N/A[4]	N/A[4]	**67.5%±3.3%**
GINE-ϵ	80.7%±5.0%	**83.26%±0.91%**	76.0%±2.2%
GINE-ϵ-JK	**84.2%±3.0%**	82.71%±0.93%	**79.1%±2.2%**
NERO$_{A,V,R}$	**94.2%±4.2%**	**85.2%±2.8%**	**86.6%±4.1%**
NERO$_{A,V,\varnothing}$	88.5%±6.9%	83.7%±4.4%	85.1%±3.4%
NERO$_{\varnothing,V,R}$	66.3%±5.9%	65.4%±5.1%	70.5%±8.2%

paradigm. Second, when completion time is still an issue, one can sample a smaller vertex subset and calculate only the associated embedding fragments, resulting in an approximate solution. Finally, if there is any information associated with individual embedding elements available (e.g. feature importance scores), it can be traced back to the network components that contributed to that particular cell. Figure 3b shows a proof of concept of that application — a venation graph is painted using a heatmap obtained from the impurity values provided by an extra-trees classifier.



(a) Venation fragment. (b) Importance heatmap.

Fig. 3: Visualisation of element importances based on extra-trees impurity scores.

6 Conclusions and future works

The presented embedding technique performed similarly or better than all the other graph kernels it was compared to on all but one of the utilised benchmark datasets. Obtained accuracies were often on-par with general-purpose GNN methods, without the need for costly GPU-based learning computations. This property was especially useful in case of large laminae venation graphs, where many of those models stopped being applicable due to memory constraints. The technique itself is highly modular and could be further adapted to the better suit a given problem. Lastly, its outcomes can be directly associated with the original nodes and edges, facilitating any attempts at results explanation.

Nevertheless, there are still topics requiring additional studies. The embedding may be further enriched by incorporating supplementary information about

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_18

the structure itself, such as node centrality measures. Advanced trait digitisation schemes can be designed utilising tiling algorithms, clustering procedures, and fuzzy boundaries. Finally, the real potential of the method interpretability should be assessed in a separate investigation.

Acknowledgements The authors would like to express their sincerest gratitude to Jana Lasser for providing a unique venation dataset and for her useful tips about its usage.

References

1. Al-Rfou, R., Perozzi, B., Zelle, D.: Ddgk: Learning graph representations for deep divergence graph kernels. In: The World Wide Web Conference. pp. 37–48 (2019)
2. Bagrow, J.P., Bollt, E.M., Skufca, J.D., Ben-Avraham, D.: Portraits of complex networks. EPL (Europhysics Letters) **81**(6), 68004 (2008)
3. Borgwardt, K.M., Kriegel, H.P.: Shortest-path kernels on graphs. In: Fifth IEEE international conference on data mining (ICDM'05). pp. 8–pp. IEEE (2005)
4. Bunke, H., Riesen, K.: Towards the unification of structural and statistical pattern recognition. Pattern Recognition Letters **33**(7), 811–825 (2012)
5. Chang, C.C., Lin, C.J.: Libsvm: A library for support vector machines. ACM transactions on intelligent systems and technology (TIST) **2**(3), 1–27 (2011)
6. Czech, W.: Invariants of distance k-graphs for graph embedding. Pattern Recognition Letters **33**(15), 1968–1979 (2012)
7. Czech, W., Łazarz, R.: A method of analysis and visualization of structured datasets based on centrality information. In: International Conference on Artificial Intelligence and Soft Computing. pp. 429–441. Springer (2016)
8. Czech, W., Mielczarek, W., Dzwinel, W.: Distributed computing of distance-based graph invariants for analysis and visualization of complex networks. Concurrency and Computation: Practice and Experience **29**(9), e4054 (2017)
9. Debnath, A.K., Lopez de Compadre, R.L., Debnath, G., Shusterman, A.J., Hansch, C.: Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. Journal of medicinal chemistry **34**(2), 786–797 (1991)
10. Diehl, F.: Edge contraction pooling for graph neural networks. CoRR (2019)
11. Dobson, P.D., Doig, A.J.: Distinguishing enzyme structures from non-enzymes without alignments. Journal of molecular biology **330**(4), 771–783 (2003)
12. Gao, H., Ji, S.: Graph u-nets. In: International Conference on Machine Learning. pp. 2083–2092 (2019)
13. Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. Machine learning **63**(1), 3–42 (2006)
14. Hacibeyoglu, M., Ibrahim, M.H.: Ef_unique: an improved version of unsupervised equal frequency discretization method. Arabian Journal for Science and Engineering **43**(12), 7695–7704 (2018)
15. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in neural information processing systems. pp. 1024–1034 (2017)
16. Jiang, J., Cui, Z., Xu, C., Yang, J.: Gaussian-induced convolution for graphs. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 4007–4014 (2019)
17. Kriege, N.M., Giscard, P.L., Wilson, R.: On valid optimal assignment kernels and applications to graph classification. Advances in Neural Information Processing Systems **29**, 1623–1631 (2016)

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_18

18. Kuhn, F., Schneider, P.: Computing shortest paths and diameter in the hybrid network model. In: Proceedings of the 39th Symposium on Principles of Distributed Computing. pp. 109–118 (2020)
19. Maron, H., Ben-Hamu, H., Serviansky, H., Lipman, Y.: Provably powerful graph networks. In: Advances in neural information processing systems. pp. 2156–2167 (2019)
20. Morris, C., Kriege, N.M., Bause, F., Kersting, K., Mutzel, P., Neumann, M.: Tudataset: A collection of benchmark datasets for learning with graphs. arXiv preprint arXiv:2007.08663 (2020)
21. Neumann, M., Garnett, R., Bauckhage, C., Kersting, K.: Propagation kernels: efficient graph kernels from propagated information. Machine Learning **102**(2), 209–245 (2016)
22. Nguyen, D., Nguyen, T., Phung, D.: Universal self-attention network for graph classification. arXiv preprint arXiv:1909.11855 (2019)
23. Niepert, M., Ahmed, M., Kutzkov, K.: Learning convolutional neural networks for graphs. In: International conference on machine learning. pp. 2014–2023 (2016)
24. Rapacz, M., Łazarz, R.: Automatic extraction of leaf venation complex networks. In: ECAI 2020. vol. 325, p. 1914–1921. IOS Press (2020)
25. Ronellenfitsch, H., Lasser, J., Daly, D.C., Katifori, E.: Topological phenotypes constitute a new dimension in the phenotypic space of leaf venation networks. PLoS computational biology **11**(12), e1004680 (2015)
26. Roth-Nebelsick, A., Uhl, D., Mosbrugger, V., Kerp, H.: Evolution and function of leaf venation architecture: a review. Annals of Botany **87**(5), 553–566 (2001)
27. Shervashidze, N., Schweitzer, P., Van Leeuwen, E.J., Mehlhorn, K., Borgwardt, K.M.: Weisfeiler-lehman graph kernels. Journal of Machine Learning Research **12**(9) (2011)
28. Shervashidze, N., Vishwanathan, S., Petri, T., Mehlhorn, K., Borgwardt, K.: Efficient graphlet kernels for large graph comparison. In: Artificial Intelligence and Statistics. pp. 488–495 (2009)
29. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
30. Togninalli, M., Ghisu, E., Llinares-López, F., Rieck, B., Borgwardt, K.: Wasserstein weisfeiler-lehman graph kernels. In: Advances in Neural Information Processing Systems. pp. 6439–6449 (2019)
31. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: International Conference on Learning Representations (2018)
32. Wale, N., Watson, I.A., Karypis, G.: Comparison of descriptor spaces for chemical compound retrieval and classification. Knowledge and Information Systems **14**(3), 347–375 (2008)
33. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? In: International Conference on Learning Representations (2018)
34. Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.i., Jegelka, S.: Representation learning on graphs with jumping knowledge networks. In: International Conference on Machine Learning. pp. 5453–5462 (2018)
35. Zhang, Z., Bu, J., Ester, M., Zhang, J., Yao, C., Yu, Z., Wang, C.: Hierarchical graph pooling with structure learning. arXiv preprint arXiv:1911.05954 (2019)
36. Zhang, Z., Cui, P., Zhu, W.: Deep learning on graphs: A survey. IEEE Transactions on Knowledge and Data Engineering (2020)

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_18