ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

Deep Learning for Graphs

Davide Bacciu[1], Federico Errica[2], Nicolò Navarin[3], Luca Pasa[3] and Daniele Zambon[4]

1- Department of Computer Science, University of Pisa, Italy

2- NEC Laboratories Europe, Germany

3- Department of Mathematics, University of Padova, Italy

4- The Swiss AI Lab IDSIA & Università della Svizzera italiana, Switzerland

Abstract. The flourishing field of deep learning for graphs relies on the layered computation of representations from graph-structured input data. Message passing is the most common strategy for such processing of graphs, based on an efficient information exchange among the connected nodes via a local and iterative procedure. Representations learned in this way can be used to address different tasks related to nodes, edges, or even entire graphs. This tutorial paper reviews fundamental concepts and open challenges of deep learning for graphs and summarizes the contributions that have been accepted for publication to the ESANN 2022 special session on the topic.

1 Introduction

In this tutorial paper, we present the basic concepts of the relatively recent research topic of deep learning for graphs, which is the subject of a special session at the 30^{th} European Symposium on Artificial Neural Networks, Computational Intelligence, and Machine Learning organized by the authors.

Deep learning for graphs is a flourishing field that encompasses machine learning models, called Deep Graph Networks (DGNs) [1], composed of several abstraction layers and addressing tasks associated with graph-structured data. Generally speaking, a graph is an object that represents entities interacting with each other. Both the entities and their relations — usually called *nodes* and *edges*, respectively — can have information attached to them, like atom and bond types in chemical compounds, allowing us to flexibly represent many real-world phenomena. A pictorial example of a molecular graph is shown in Figure 1 to ease the subsequent exposition. The addressed tasks can be associated with nodes, edges, or properties of entire graphs. An example of task over nodes is the prediction of properties of a social network user based on her/his connections. In this setting, the dataset usually consists of a single large graph and the observations are the nodes of the graph with the associated information. Similarly, suggesting relationships (e.g., friendships) in a social network can be cast as an edge-level prediction task. In graph-level prediction settings, instead, each input is a whole graph and the goal is inferring associated (global) properties, e.g., toxicity in humans of chemical compounds.

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

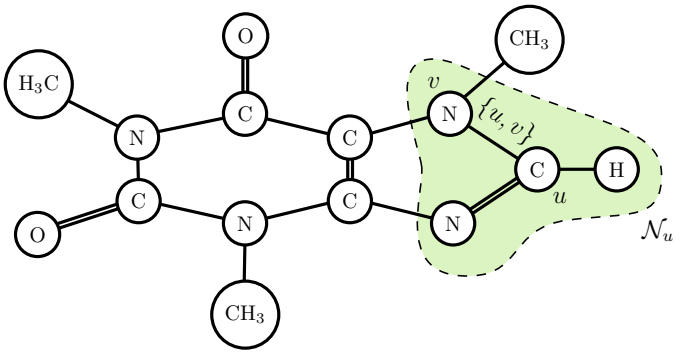

Fig. 1: Caffeine molecule represented as a graph where nodes denote atoms/groups with their type and edges reflect whether the bond is single or double. The neighborhood $\mathcal{N}(u)$ of node u is marked by the green dashed area.

2 Basic Definitions and Notation

We introduce some basic terminology about graphs that will be useful throughout the document. A graph can be defined as a tuple $g = (\mathcal{V}, \mathcal{E}, \mathcal{X}, \mathcal{A})$ where $\mathcal{V}$ represents the set of nodes (also known as vertices) and $\mathcal{E}$ defines the edges (or arcs) representing interactions between pairs of nodes. When we deal with **undirected** graphs, edges encode information about *unordered* pairs, i.e., $\mathcal{E} \subseteq \{\{u, v\} \mid u, v \in \mathcal{V}\}$; in contrast, a **directed** graph specifies ordered pairs of interactions between nodes, i.e., $\mathcal{E} \subseteq \{(u, v) \mid u, v \in \mathcal{V}\}$. As a way of comparison, the graph in Figure 1 is undirected. The sets $\mathcal{X}$ and $\mathcal{A}$ define the domain of the features attached to nodes and edges, respectively; the most typical case is to use $\mathbb{R}^{d_e}, d_e \in \mathbb{N}^+$ as the domain of node features and some finite alphabet for discrete types of relationships.

There are different ways in which we can represent edge information. One of these is by an *adjacency matrix* $\mathbf{A} \in \{0, 1\}^{|\mathcal{V}| \times |\mathcal{V}|}$. Assuming $\mathcal{V} = \{1, 2, \ldots, n\}$, a generic entry u, v of $\mathbf{A}$ is 1 when $(u, v) \in \mathcal{E}$ (directed case) or $\{u, v\} = \{v, u\} \in \mathcal{E}$ (undirected case), and it is 0 otherwise; therefore, adjacency matrices of an undirected graphs are symmetric. More generally, the same n-node graph can be represented by multiple, yet all equivalent, adjacency matrices arising from all $n!$ possible indexing of the nodes in $\mathcal{V}$. Such ambiguity gives rise to what is known as *graph isomorphism problem* [2].

In **dense** graphs, most of the entries of $\mathbf{A}$ are one, whereas the opposite is true in **sparse** graphs. For large and sparse graphs, the adjacency matrix can become an inefficient representation of the information, as most of the entries are 0; in this case, a more convenient representation is an adjacency list, that specifies, for each node, which are its *neighbors*. The neighborhood of node v is defined as the set of nodes directly connected to it: $\mathcal{N}_v = \{u \in \mathcal{V} \mid (u, v) \in \mathcal{E}\}$. When the neighborhood always includes the node v itself, we speak of "closed" neighborhood ("open" otherwise). The neighborhood of node u is depicted by

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1.
Available from http://www.i6doc.com/en/.

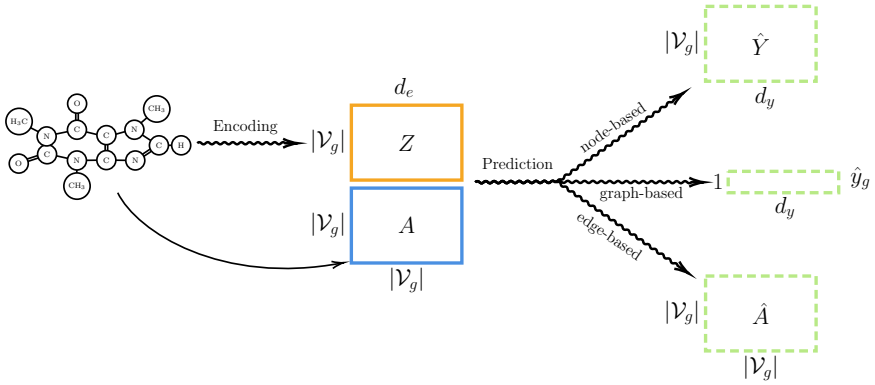

Fig. 2: The general framework for learning on graphs. First, the input graph is encoded and latent node representations Z are produced, then standard machine learning is applied to generate the desired output. Node (edge) tasks require one prediction per node (edge), whereas in graph-level tasks the node representations are aggregated to produce a single output for the whole graph.

the green dashed area in Figure 1. We conclude the section by introducing the concept of permutation invariance, which is extremely useful in all the methods presented below, including those that contributed to the special session. A function is said permutation invariant if it does not change its output when the components of the input are reordered according to some permutation. The sum, the mean, the maximum and the product operators are straightforward examples of permutation invariant functions. In the context of graphs, permutation invariance is intended with respect to the permutation of the nodes.

3 The Building Blocks

Dealing with structured data often requires defining machine-learning models able to learn *graph isomorphic invariant* representations, or embeddings, for nodes, thus producing graphs with the same topology of the input graph but different node features. This isomorphic transduction of the graph allows to tackle nodes, edges, and graph-related tasks; for instance, a graph representation can be easily computed by aggregating together its nodes' representations. Node/graph embeddings are usually learned only as intermediate representations to solve a downstream task, but it is not uncommon to use such embeddings for other purposes, e.g., data visualization. For instance, we may compute the distance or kernel between two node embeddings to assess the affinity of two corresponding users in a social network, or jointly train a graph-level embedding module and a dense feed-forward neural network to predict a drug's toxicity.

The Message-passing paradigm. The vast majority of DGNs rely on the message-passing processing paradigm [3] to propagate information between neighboring

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

nodes. The message-passing paradigm provides easy-to-implement operations that allow subsequent refinements of the node embeddings. Typically, a message-passing operation can be written as

$$\mathbf{h}_v^{\ell+1} = \text{Update}\Big(\mathbf{h}_v^{\ell}\,,\ \text{Aggregate}\big(\ \{\!\{\mathbf{h}_u^{\ell} : u \in \mathcal{N}(v)\}\!\}\ \big)\Big), \qquad \forall\, v \in \mathcal{V}, \tag{1}$$

where $\mathbf{h}_v^{\ell} \in \mathbb{R}^{d_e}$ denotes a d_e-dimensional embedding of generic node v at layer ℓ, $\mathbf{h}_v^{\ell+1}$ the updated embedding after message passing, and $\{\!\{\cdot\}\!\}$ denotes a multiset; more general forms, accounting for edge information can be defined as well [1]. Stacking several message-passing layers results in typical DGNs, such as those by [4, 5, 6, 7, 8]. The advantages of operations like (1) are manifold, including the fact that the operations are performed locally in the graphs (so that updating a node does not require processing all graph nodes) and that the parametrization of Update and Aggregate functions does not depend on the number of neighboring nodes and can be applied to graphs of different sizes and topologies. The message-passing operation in (1) at layer ℓ can be also represented in matrix form in most cases; a simple example is

$$\mathbf{H}^{\ell+1} = \mathbf{H}^{\ell} \,\|\, \mathbf{A}\mathbf{H}^{\ell}\mathbf{W}^{\ell}, \tag{2}$$

where $\mathbf{H}^{\ell} = [\dots|\mathbf{h}_v^{\ell}|\dots]^{\top}$ is the matrix stack of all input node representations and $\mathbf{H}^{\ell+1}$ that of the output. Confronting equations (1) and (2), $\text{Update}(a, b)$ in (1) is instantiated in (2) as $a \| b$, that is the concatenation function, and Aggregate is instantiated as the sum of the node embeddings transformed by a linear map parametrized by a learnable matrix $\mathbf{W}^{\ell} \in \mathbb{R}^{d_e \times d_e}$. Moreover, note that message-passing layers as in (1) and (2) are invariant to node permutations by construction: for any permutation matrix $\mathbf{\Pi}$, the output of (2) when feeding $\mathbf{\Pi}\,\mathbf{H}^{\ell}$ as input is $\mathbf{\Pi}\,\mathbf{H}^{\ell+1}$. Efficient implementations nowadays rely on sparse matrix multiplications [9, 10]. Another path that has been recently pursued to define efficient architectures is reducing the excess complexity of common DGNs by removing the non-linearities [11, 12]. In addition, Reservoir Computing (RC) approaches that provide a way to implement extremely efficient alternatives to end-to-end training of Neural Networks have been extended to graph domains [13, 14, 15]. We also remark that message passing is not limited to neural architectures: fully unsupervised, deep and probabilistic models exist as well [16, 17, 18]. Finally, we mention that DGNs following different computational paradigms have been proposed as well (for example, see [19]).

Expressive power of DGNs. Not all DGNs are equally powerful in solving given problems. The most common strategy to assess the expressive power relies on the Weisfeiler-Lehman (WL) method [20] which results in a hierarchy of tests of increasing power in deciding whether or not two graphs are isomorphic; several papers relied on the WL criterion [8, 21, 19, 22], with [23] proving universal approximation capabilities for some DGNs. Nevertheless, some limitations have started emerging, and other strategies have gained attention [24, 25, 26, 27]. From a machine learning perspective, other aspects matter too, like the ability to produce similar embeddings, or outputs, when processing similar inputs [28, 29].

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

Graph Pooling. Similarly to pooling in convolutional neural networks for regular grids, graph convolutional layers are often interleaved with graph pooling layers. Graph pooling operators, however, have been used also for graph coarsening. Performing pooling in the context of graphs turns out to be very complex mainly for the lack of a predefined structure, like grid connectivity. In order to address these issues, in the last few years, different types of graph pooling operators have been proposed [30, 31, 32, 33, 34, 35, 36]. Virtually every pooling operator can be implemented as the composition of three operations: Select, Reduce, and Connect, as highlighted by [37], which helps to analyze and guide the choice of an appropriate pooling method to use in the given application.

4 What Lies Ahead

In this section, we give pointers to some of the most promising and exciting directions in the field of deep learning that we see in the literature.

Uncertainty Quantification. DGNs, as many other machine learning methods, generally lack the ability to specify some measure of *confidence* about their predictions, let the problem be classification or regression [38]. This is a serious problem when the predictions are used in high-stake domains like healthcare, e.g., patient networks. By endowing DGNs with sound ways to estimate their uncertainty, the end user can make better informed decisions about the next course of action.

Spatiotemporal graphs. In applications like those involving sensor and transportation networks, we deal with time series displaying both temporal dependencies as well as dependencies among the entities that are producing the time series. In such settings, promising DGNs for such spatiotemporal graphs have appeared in the literature to tackle problems such as time series forecasting and missing-data imputation, e.g., see the works by [39, 40, 41].

Structure Learning. The information carried by the structure of graphs is often useful to improve the predictive performance compared to structure-agnostic alternatives. On the other hand, by *learning the graph structure* we can discover additional interactions which may have been missing in the original graph [42] or generate entirely new chemical compounds with specific properties of interest [43]. That said, the most common structure learning methods scale poorly to large datasets, as they try to predict the entire adjacency matrix of the graph. Investigation of more efficient ways to do so may pave the way for large-scale structure learning.

5 Special Session's Contributions

The contributions to this special session address a heterogeneous and quite original set of problems that can be formulated as a graph learning task.

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

- Romero and De Bie [44] propose a new method to recommend songs to users by exploiting a dataset of playlists represented as a weighted graph. The authors frame the problem as a link prediction task and derive a graph-embedding strategy, based on a Poisson model, to predict the continuous edge weight between two songs. The strategy is shown to be more effective than common song-recommendation baselines without significant overhead.
- Tortorella and Micheli [45] evaluate Graph Echo State Networks (GESN) [13] on semi-supervised node classification tasks with varying degrees of homophily (i.e., the ratio of connected nodes with the same value for a target attribute). The goal is to analyze the impact of oversmoothing of representations, a problem known to severely impact the performances of popular models like the Graph Convolutional Network (GCN) [6]. The empirical analysis provides a favorable comparison of GESN compared to ad-hoc architectural variations that counteract oversmoothing.
- Saveri and Bortolussi [46] investigate the problem of propositional model counting (#SAT) using a combination of Belief Propagation and Graph Attention Network (GAT) [7]. The learning problem is formulated as a factor graph and decomposed as a product of functional terms, which are implemented using the attention mechanism of GAT. The authors show that the approach scales to Boolean formulae of larger sizes and generalizes to others with different distributions of terms.
- Landolfi [47] develops a parallel algorithm for the efficient computation of EdgePool [48], a sparse pooling method that, unlike other alternatives, preserves the connectivity of the original graph structure by performing a series of edge contractions. By leveraging well-known reduction techniques from graph theory, the proposed method can outperform other competitors while scaling to larger graphs, thus making EdgePool a widely applicable technique.
- Finally, Caldart *et al.* [49] address the fairness problem from the perspective of bias being introduced by the relational information in the graph. The authors extend an existing perturbation-based fairness method, proposing to perturb the adjacency matrix in a biased way to reduce the amount of edge homophily in the graph; then they exploit an appropriate loss function to learn fair representations. The approach is shown to improve over all fairness metrics on different node classification tasks.

6 Conclusions

Deep learning for graphs is a flourishing research field with an increasing number of published papers every year. With this tutorial and ESANN 2022 special session, we gave an overview of the most common building blocks for processing graphs, like message-passing and pooling layers. We reported some of the most

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

promising research directions that have attracted recent attention. Finally, we summarized the interesting papers contributing to the present special session.

References

[1] Davide Bacciu, Federico Errica, Alessio Micheli, and Marco Podda. A gentle introduction to deep learning for graphs. *Neural Networks*, 129:203–221, 9 2020.

[2] Johannes Kobler, Uwe Schöning, and Jacobo Torán. *The graph isomorphism problem: its structural complexity*. Springer Science & Business Media, 2012.

[3] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural Message Passing for Quantum Chemistry. In *ICML*, pages 1263—-1272, apr 2017.

[4] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[5] Alessio Micheli. Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*, 20(3):498–511, 2009.

[6] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[7] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

[8] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.

[9] Matthias Fey and Jan Eric Lenssen. Fast Graph Representation Learning with PyTorch Geometric, 5 2019.

[10] Daniele Grattarola and Cesare Alippi. Graph neural networks in tensorflow and keras with spektral [application notes]. *IEEE Computational Intelligence Magazine*, 16(1):99–106, 2021.

[11] Felix Wu, Tianyi Zhang, Amauri Holanda de Souza, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. Simplifying Graph Convolutional Networks. *ICML*, 2019.

[12] Nicolò Navarin, Wolfgang Erb, Luca Pasa, and Alessandåro Sperduti. Linear Graph Convolutional Networks. In *ESANN*, 2020.

[13] Claudio Gallicchio and Alessio Micheli. Graph echo state networks. In *IJCNN*, pages 1–8. IEEE, 2010.

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

[14] Claudio Gallicchio and Alessio Micheli. Fast and deep graph neural networks. In *AAAI*, 2020.

[15] Luca Pasa, Nicolò Navarin, and Alessandro Sperduti. Multiresolution reservoir graph neural network. *IEEE Transactions on Neural Networks and Learning Systems*, 33(6):2642–2653, 2022.

[16] Davide Bacciu, Federico Errica, and Alessio Micheli. Contextual Graph Markov Model: A deep and generative approach to graph processing. In *ICML*, volume 80, pages 294–303, 2018.

[17] Davide Bacciu, Federico Errica, and Alessio Micheli. Probabilistic learning on graphs via contextual architectures. *Journal of Machine Learning Research*, 21(134):1–39, 2020.

[18] Daniele Castellana, Federico Errica, Davide Bacciu, and Alessio Micheli. The infinite contextual graph Markov model. In *ICML*, pages 2721–2737, 2022.

[19] Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. *NeurIPS*, 32, 2019.

[20] Boris Weisfeiler and Andrei Leman. The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Series*, 2(9):12–16, 1968.

[21] Zhengdao Chen, Soledad Villar, Lei Chen, and Joan Bruna. On the equivalence between graph isomorphism testing and function approximation with gnns. *NeurIPS*, 32, 2019.

[22] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *AAAI*, volume 33, pages 4602–4609, 2019.

[23] Nicolas Keriven and Gabriel Peyré. Universal invariant and equivariant graph neural networks. *NeurIPS*, 32, 2019.

[24] Nils M Kriege, Christopher Morris, Anja Rey, and Christian Sohler. A property testing framework for the theoretical expressivity of graph kernels. In *IJCAI*, pages 2348–2354, 2018.

[25] Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Approximation ratios of graph neural networks for combinatorial problems. *NeurIPS*, 32, 2019.

[26] Andreas Loukas. What graph neural networks cannot learn: depth vs width. In *ICLR*, 2020.

[27] Giorgos Bouritsas, Fabrizio Frasca, Stefanos P Zafeiriou, and Michael Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

[28] Nils M Kriege, Fredrik D Johansson, and Christopher Morris. A survey on graph kernels. *Applied Network Science*, 5(1):1–42, 2020.

[29] Daniele Zambon, Cesare Alippi, and Lorenzo Livi. Graph random neural features for distance-preserving graph representations. In *ICML*, Proceedings of Machine Learning Research, pages 10968–10977, 2020.

[30] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11):1944–1957, 2007.

[31] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *NeurIPS*, pages 4800–4810, 2018.

[32] Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. Hierarchical representation learning in graph neural networks with node decimation pooling. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[33] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Spectral clustering with graph neural networks for graph pooling. In *ICML*, pages 874–883. PMLR, 2020.

[34] Davide Bacciu and Luigi Di Sotto. A non-negative factorization approach to node pooling in graph convolutional neural networks. In *AIxIA*, pages 294–306. Springer, 2019.

[35] Davide Bacciu, Alessio Conte, Roberto Grossi, Francesco Landolfi, and Andrea Marino. K-plex cover pooling for graph neural networks. *Data Mining and Knowledge Discovery*, 35(5):2200–2220, 2021.

[36] Luca Pasa, Nicolò Navarin, and Alessandro Sperduti. Som-based aggregation for graph convolutional neural networks. *Neural Computing and Applications*, 34(1):5–24, 2022.

[37] Daniele Grattarola, Daniele Zambon, Filippo Bianchi, and Cesare Alippi. Understanding pooling in graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–11, 2022.

[38] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, et al. A survey of uncertainty in deep neural networks. *arXiv preprint arXiv:2107.03342*, 2021.

[39] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling. In *IJCNN*, pages 1907–1913, 2019.

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

[40] Daniele Zambon and Cesare Alippi. Az-whiteness test: a test for uncorrelated noise on spatio-temporal graphs, 2022.

[41] Andrea Cini, Ivan Marisca, and Cesare Alippi. Filling the g_ap_s: Multivariate time series imputation by graph neural networks. In *ICLR*, 2022.

[42] Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. Learning discrete structures for graph neural networks. In *ICML*, pages 1972–1982. PMLR, 2019.

[43] Marco Podda. Deep learning on graphs with applications to the life sciences. *PhD Thesis*, 2021.

[44] Raphaël Romero and Tijl De Bie. Embedding-based next song recommendation for playlists. In *ESANN*, 2022.

[45] Domenico Tortorella and Alessio Micheli. Beyond homophily with graph echo state networks. In *ESANN*, 2022.

[46] Gaia Saveri and Luca Bortolussi. Graph neural networks for propositional model counting. In *ESANN*, 2022.

[47] Francesco Landolfi. Revisiting edge pooling in graph neural networks. In *ESANN*, 2022.

[48] Frederik Diehl, Thomas Brunner, Michael Truong Le, and Alois Knoll. Towards graph pooling by edge contraction. In *ICML 2019 workshop on learning and reasoning with graph-structured data*, 2019.

[49] Federico Caldart, Luca Pasa, Luca Oneto, Alessandro Sperduti, and Nicolò Navarin. Biased edge dropout in nifty for fair graph representation learning. In *ESANN*, 2022.