ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2. Available from http://www.i6doc.com/en/.

Towards the Application of Backpropagation-Free Graph Convolutional Networks on Huge Datasets

Nicolò Navarin[1], Luca Pasa[1] and Alessandro Sperduti[1] *

1- University of Padua - Department of Mathematics
via Trieste 63, 35121 Padua - Italy

Abstract. Backpropagation-Free Graph Convolutional Networks (BF-GCN) are backpropagation-free neural models dealing with graph data based on Gated Linear Networks. Each neuron in a BF-GCN is defined as a *set* of graph convolution filters (weight vectors) and a gating mechanism that, given a node's *context*, selects the weight vector to use for processing the node's attributes based on its distance from a set of prototypes. Given the higher expressivity BF-GNN's neurons compared to the standard graph convolutional neural networks' ones, they show bigger memory footprint. In this paper, we explore how reducing the size of node contexts through randomization can reduce the memory occupancy of the method, enabling its application to huge datasets. We empirically show how working with very low dimensional contexts does not impact the resulting predictive performances.

1 Introduction

In the last years, the field of learning from graph data [1, 2] has seen significant development, with Graph Neural Networks (GNNs) emerging as the preferred model for tackling graph-related problems.Backpropagation-Free Graph Convolutional Networks (BF-GCNs) [3] have been recently proposed and shown to provide predictive performances comparable to their backpropagation-based counterparts. In contrast to the widespread GNNs, BF-GCNs present an alternative paradigm in the realm of graph neural networks, wherein the training process does not rely on backpropagation. Each neuron in a BF-GCN is based on a gating mechanism that divides the *context space* into regions. The model then learns, for each region, a linear classifier. This mechanism allows training each neuron independently, without using back-propagation, resulting in a

*We acknowledge the support of the projects: "Future AI Research (FAIR) - Spoke 2 Integrative AI - Symbolic conditioning of Graph Generative Models (SymboliG)" funded by the European Union under the National Recovery and Resilience Plan (NRRP), Mission 4 Component 2 Investment 1.3 - Call for tender No. 341 of March 15, 2022 of Italian Ministry of University and Research – NextGenerationEU, Code PE0000013, Concession Decree No. 1555 of October 11, 2022 CUP C63C22000770006; "iNEST: Interconnected Nord-Est Innovation Ecosystem" funded under the NRRP, Mission 4 Component 2 Investment 1.5 - Call for tender No. 3277 of 30 December 2021 of Italian Ministry of University and Research – NextGenerationEU, Code ECS00000043, Concession Decree No. 1058 of June 23, 2022, CUP C43C22000340006; the PON R&I 2014-2020 project *Smart Waste Treatment* founded by the FSE REAC-EU; the project "Lifelong Learning on large-scale and structured data" funded by the cloud adoption mini-competition within the OCRE framework.

ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2. Available from http://www.i6doc.com/en/.

set of convex problems to solve. Different versions of BF-GCNs have been defined [4]. One of the simplest and most effective instantiations defines regions based on prototypes (initialized in a data-driven way), where each input context is assigned to the closest prototype, obtaining a Voronoi tessellation of the context space. The resulting model is as expressive as its backpropagation-based counterpart, while offering a notably simplified training phase. Furthermore, BF-GCNs are more parallelizable, as each neuron can be updated independently after the forward pass. These features make BF-GCNs suitable for application across various contexts, particularly on extensive datasets and complex tasks. On the flip side, one of their main limitations when increasing the number of regions of each neuron, is that the network requires a significant amount of memory. This limitation primarily stems from the gating mechanism embedded in each neuron, specifically concerning the amount of memory needed to handle the mechanism that shatter the context space. Indeed the storage requirements are, for each region, the weight vector, and the prototype.

In this paper, we explore a simple method to significantly reduce the dimensionality of the prototypes affecting the memory occupancy (approximately halving the memory footprint of the model), while slightly improving model efficiency and not impacting the predictive abilities of BF-GCNs. We propose a specific simplification of the prototype-based gating mechanism, which relies on mapping the context and prototypes to a smaller space, with the aim of effectively reducing the memory demands of individual neurons within the model.

We experimentally show that, on datasets of increasing size, the predictive performances of baseline GCNs and BF-GCNs are comparable. Moreover, we show that the proposed improvement can provide improved predictive performance while reducing the memory required to run a model.

2 Backpropagation-Free Graph Convolutional Networks

A learning problem on a graph can be formulated as learning a function mapping nodes to labels. The graph structure is given as $G = (V, E, \mathcal{L})$, where $V = \{v_1, \ldots, v_n\}$ is the set of nodes, $E \subseteq V \times V$ is the set of edges, and $\mathcal{L} : \mathcal{V} \to \mathbb{R}^s$ is a function associating a vector of attributes to each node. We denote as $\mathcal{N}(v) = \{u \,|\, (v, u) \in E\}$ the set of neighbors of a node v. To simplify the notation, we define for a fixed graph G the matrix $\mathbf{X} = [\mathcal{L}(v_1), \ldots, \mathcal{L}(v_n)]^\intercal$. Given a graph G, our training set is composed by the target information associated with some of the graph nodes, i.e., $\{(v, y) \,|\, v \in W, y \in \mathcal{Y}\}$ with $W \subset V$. For the sake of simplicity, in our presentation, we will consider binary labels $\mathcal{Y} \in \{0, 1\}$.

The definition of BF-GCNs in based on Gated Linear Networks (GLNs) [5]. The objective of GLNs is to construct a model comprising neurons that can be trained locally, independently, and solely through task supervision. Each neuron acts as an independent classifier, which can be trained separately from the rest of the network given the input [6]. Recently, GLNs have been extended to graph structured data [3], reminiscent of the definition of graph convolution [7, 8]. The authors propose a generalization of the GLN mechanism in which the network

ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2. Available from http://www.i6doc.com/en/.

architecture reflects the structure of the input graph. Node representations are refined at each layer based on the local graph topology through an aggregation operation over neighboring nodes. Specifically the authors introduce the backpropagation-free version of GCN [9] which leads to the following definition of a hidden layer of BF-GCN:

$$\mathbf{H}^{(i)}_{(\mathbf{Z})} = \sigma\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}(\mathbf{I}+\mathbf{A})\tilde{\mathbf{D}}^{-\frac{1}{2}}\,\sigma^{-1}\left(\mathbf{H}^{(i-1)}_{(\mathbf{Z})}\right)\mathbf{W}^{(i)}_{(\mathbf{Z})}\right).$$

where $\mathbf{W}^{(i)}_{(\mathbf{Z})} = [\mathbf{w}^{(i)}_{1,(\mathbf{z}_1)}, \ldots, \mathbf{w}^{(i)}_{n,(\mathbf{z}_n)}]$.

In this paper, we focus on the gating mechanism utilized in the formulation above, specifically the prototype-based approach introduced in [6]. Let us define the *context* $\mathbf{z}$ associated to a node v as a vector, that can be for instance fixed to the node label $\mathcal{L}_v$ or can be the hidden representation computed at the preceding layer for the same node. The gating mechanism divides the input nodes in subsets based on their *context*, splitting the *context* space in regions. Each node will be mapped to the region its context belongs to. Each region will be associated to an independent weight vector, that will be learned during training. Each region is characterized by a prototype, and each point in the space is assigned to the region with the closest prototype, resulting in a Voronoi tessellation. Let us consider a matrix $\mathbf{P}^{(i)}_j \in \mathbb{R}^{k\times d^{(z)}}$ of prototypes associated to the j-th neuron of layer i. The context vector selection function $\mathbf{c}^{(i)}_{(\mathbf{z})} \in \{0,1\}^k$ can be formulated as: $\mathbf{c}^{(i)}_{j,(\mathbf{z})} = \text{one\_hot}(\arg\min_l(||\mathbf{p}^{(i)}_{j,l} - \mathbf{z}||)$, where $\mathbf{p}^{(i)}_{j,l}$ is the l-th row of $\mathbf{P}^{(i)}_j$ and $||\cdot||$ is the 2-norm assessing the distance between $\mathbf{p}^{(i)}_{j,l}$ and context $\mathbf{z}$. Given the pair $(\mathbf{x},\mathbf{z})$ as input, we select the weights of a single neuron j at the i-th layer (i.e. the j-th row of $\mathbf{W}^{(i)}_{(\mathbf{z})}$) as: $\mathbf{w}^{(i)}_{j,(\mathbf{z})} = \mathbf{W}^{(i)}_j \mathbf{c}^{(i)}_{j,(\mathbf{z})}$. Notice that the main characteristic of a Gated Linear Neuron is that, instead of having a single weight vector, each GL neuron depends on a *matrix* of parameters $\mathbf{W}^{(i)}_j \in \mathbb{R}^{d_{i-1}\times k}$. When defining the prototypes, it is crucial to consider the data distribution to ensures that each prototype will lie on the input data manifold. A possible solution is to initialize the gating mechanism in a data-driven way [3]. The policy consists in sampling prototypes uniformly from the training set $D^{(Tr)}$, i.e. $\mathbf{p}^{(i)}_{j,l} \sim \mathcal{U}(D^{(Tr)})$ where $\mathbf{p}^{(i)}_{j,l}$ is the l-th row of $\mathbf{P}^{(i)}_j$ and $\mathcal{U}(S)$ denotes the uniform distribution over the elements of a finite set S.

3 Reducing context dimension with random projection

The core improvement we propose in this paper is to reduce the size of the stored *context* vectors, thus reducing the memory footprint of the BF-GCN model. This slight modification can provide an improvement in the model's predictive performance as well. If the *contexts* belong to an high dimensional space, due to the course of dimensionality, the points essentially become uniformly distant from each other [10]. This effect can occur with only 10-15 dimensions [11]. Reducing the dimensionality of the context space can thus improve the coherence of

ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2. Available from http://www.i6doc.com/en/.

Dataset	#Classes	#Edges	#Train	#Val	#Test
Pubmed	3	88,651	1829	3944	3,944
WikiCS	10	216,123	7021	2340	2340
OGBN-prodcuts	47	61,859,140	195,922	48,981	2,204,126

Table 1: Datasets statistics. The columns #Train, #Val, and #Test report the number of nodes in the training, validation and test sets, respectively.

examples falling in the same region. The literature indicates that random projection serves as an effective dimensionality reduction technique, yielding results comparable to traditional methods like principal component analysis. Additionally, utilizing random projections incurs in low computational costs [12]. The utilization of random projection for reducing the dimensionality of data space has demonstrated its suitability across various models and applications [13]. In contrast to existing literature, our work employs dimensionality reduction not to shrink the input space, but rather to enhance the memory efficiency of BF-GCNs. In our experiments we initialize the random projection weights $\mathbf{R}^{(p)}$ using the Glorot uniform approach [14], inspired from recent randomized models on graphs [15]. We then apply the same weights to the prototype vectors $\mathbf{P}_j^{(i)}$ and to the context $\mathbf{z}$ associated to each graph node, obtaining the following context vector selection function $\mathbf{c}_{j,(\mathbf{z})}^{(i)} = \text{one\_hot}(\arg\min_l(||\mathbf{R}^{(p)}\mathbf{p}_{j,l}^{(i)} - \mathbf{R}^{(p)}\mathbf{z}||)$.

4 Results and Discussion

We empirically validated the proposed reduced-context-BF-GCN on three widely adopted datasets for node classification of increasing size: Pubmed, WikiCS [16], and OGBN-products [17]. We selected these datasets to assess the proposed approach across diverse graphs of varying size and complexity. Relevant statistics on the datasets are reported in Table 1. To asses the classification capability of the BF-GCNs we opted for the classification accuracy, since it is the most common choice for the benchmark dastsets we considered. We also evaluate the model performances in terms of memory consumption and time complexity. We performed our experiments on a G2 Google cloud machine with 12 vCPUs, 48GBs of RAM and an Nvidia L4 GPU with 24GB of video memory.

Let us start our analysis focusing on the Pubmed and WikiCS datasets. Here we limit the number of layers to 2 since it has been shown that increasing them did not result in any accuracy gain [4]. Consequently, we could validate a higher number of prototypes per neuron (from 2 up to 32), since increasing each neuron's non-linearity can be beneficial. In Tables 2a and 2b we show the performance and memory footprint of: the GCN model trained with back-propagation; the BF-GLN method (with no context dimensionality reduction applied) in the column *Full Context*; the proposed reduced-context-BF-GCN with different context sizes in the remaining columns. The proposed reduced-

ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2. Available from http://www.i6doc.com/en/.

Model/Context Size	Full Context	2	4	8	16	32
GCN (2,48,-)	$88.5_{\pm 0.3}$	-	-	-	-	-
Memory	390MB	-	-	-	-	-
BF-GCN (2,4,4)	$87.2_{\pm 0.2}$	$87.3_{\pm 0.4}$	$87.3_{\pm 0.4}$	$87.2_{\pm 0.3}$	$87.1_{\pm 0.2}$	$87.1_{\pm 0.3}$
Memory	752MB	316MB	316MB	316MB	316MB	336MB

(a) Accuracy results on the Pubmed dataset.

Model/Context Size	Full Context	2	4	8	16	32
GCN (1,64,-)	$81.6_{\pm 0.7}$	-	-	-	-	-
Memory	634 MB	-	-	-	-	-
BF-GCN (2,4,32)	$82.2_{\pm 0.8}$	$82.1_{\pm 1.0}$	$82.3_{\pm 1.0}$	$82.5_{\pm 1.0}$	$82.3_{\pm 0.5}$	$82.5_{\pm 0.7}$
Memory	175 MB	81MB	81MB	81MB	81MB	81MB

(b) Accuracy results on the WikiCS dataset.

Model/Context Size	Full Context	4	8
GCN (3,192,-)	$75.5_{\pm 0.2}$		-
Memory	21GB	-	-
BF-GCN (3,4,4)	$73.7_{\pm 0.2}$	$75.7_{\pm 0.5}$	$75.0_{\pm 0.5}$
Memory	19GB	11.9GB	12GB

(c) Accuracy results on the OGBN-products dataset.

Table 2: Accuracy on Pubmed (a), WikiCS (b) and OGBN-products (c) of the baseline GCN, BF-GCN and reduced-context-BF-GCN with different context sizes. We report the following hyperparameters: (#layers, #hidden, #contexts).

context-BF-GCN does not deteriorate the BF-GCN's predictive performances, and even extremely small context sizes give only a slight loss in accuracy. Moreover, the memory footprint of the BF-GCN is smaller compared to GCN, and it is further reduced by the proposed method. Let us now focus on the bigger OGBN-products dataset in Table 2c, in which the hyperparameters grid was more limited due to memory constraints. In fact, the best GCN model exploits almost all the available video memory. BF-GCN, with a similar memory footprint, achieves slightly lower predictive performance. Applying randomized context dimensionality reduction, we not only reduce the memory footprint, but also improve over the BF-GCN predictive performance, matching GCN's one.

Considering the computational times, we focus on OGBN-Products since for the other datasets the time required to perform a single epoch was too small to appreciate differences ($< \frac{1}{100}s$). The GCN requires on average 3.67 seconds per epoch. BF-GCN with full context requires 0.36 seconds per epoch for each class in a one-vs-rest approach. Our proposed BF-GCN with randomized contexts, considering for instance context size of 8, requires 0.27 seconds per epoch.

5 Conclusions and Future Works

In this paper, we proposed an improvement of the Backpropagation-free GCN model based on reducing the dimensionalty of context vectors. We show the pro-

ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2. Available from http://www.i6doc.com/en/.

posed modification in beneficial from the memory requirements and predictive performances points of view. As a future work, we will study a version of BF-GCN that can handle multiple classes at the same time, without resorting to the one-vs-rest approach. This strategy is aimed at reducing the number of model neurons required, thereby further diminishing the model's memory footprint.

References

[1] Alessandro Sperduti and Antonina Starita. Supervised neural networks for the classification of structures. *IEEE Trans. Neural Networks*, 8(3):714–735, 1997.

[2] F. Scarselli, M. Gori, Ah Chung Ah Chung Tsoi, M. Hagenbuchner, and G. Monfardini. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1), 2009.

[3] Luca Pasa, Nicolò Navarin, Wolfgang Erb, and Alessandro Sperduti. Backpropagation-free graph neural networks. In *2022 IEEE International Conference on Data Mining (ICDM)*, pages 388–397, 2022.

[4] Luca Pasa, Nicolò Navarin, Wolfgang Erb, and Alessandro Sperduti. A unified framework for backpropagation-free soft and hard gated graph neural networks. *Knowledge and Information Systems*, 66(4):2393–2416, April 2024.

[5] Joel Veness, Tor Lattimore, David Budden, Avishkar Bhoopchand, Christopher Mattern, Agnieszka Grabska-Barwinska, Eren Sezener, Jianan Wang, Peter Toth, Simon Schmitt, and et al. Gated linear networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(11):10015–10023, May 2021.

[6] Matteo Munari, Luca Pasa, Daniele Zambon, Cesare Alippi, and Nicoló Navarin. Understanding catastrophic forgetting of gated linear networks in continual learning. *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2022.

[7] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Neural Information Processing Systems (NIPS)*, jun 2016.

[8] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 2011.

[9] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*, pages 1–14, 2017.

[10] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, October 2012.

[11] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful?

[12] Sampath Deegalla and Henrik Bostrom. Reducing high-dimensional data by principal component analysis vs. random projection for nearest neighbor classification. In *2006 5th International Conference on Machine Learning and Applications (ICMLA'06)*, 2006.

[13] Haozhe Xie, Jie Li, and Hanqing Xue. A survey of dimensionality reduction techniques based on random projection. *arXiv preprint arXiv:1706.04371*, 2017.

[14] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, 2010.

[15] Nicolò Navarin, Luca Pasa, Luca Oneto, and Alessandro Sperduti. An empirical study of over-parameterized neural models based on graph random features. In *ESANN 2023 proceesdings*, page 17–22, Bruges (Belgium) and online, 2023. Ciaco - i6doc.com.

[16] Péter Mernyei and Cătălina Cangea. Wiki-cs: A wikipedia-based benchmark for graph neural networks. *arXiv preprint arXiv:2007.02901*, 2020.

[17] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.