ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-6327-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICASSP49357.2023.10095381

HIERARCHICAL GRAPH LEARNING FOR STOCK MARKET PREDICTION VIA A DOMAIN-AWARE GRAPH POOLING OPERATOR

Arie N. Arya, Yao Lei Xu, L. Stankovic, Danilo P. Mandic

Dept. of Electrical and Electronic Engineering, Imperial College London, London SW7 2AZ, UK
E-mails: {arie.arya18, yao.xu15, d.mandic}@imperial.ac.uk, ljubisa@ac.me

ABSTRACT

The utility of Graph Neural Networks (GNN) for the paradigm of forecasting short-term stock price movements is investigated. In particular, a finance-specific graph pooling operation, referred to as *StockPool*, is introduced to efficiently coarsen the stock graph. This is achieved by employing domain knowledge to cluster stocks, depending on some task-specific characteristics (e.g. industries, sub-industries, etc.). Unlike fully end-to-end learnable graph pooling strategies (e.g. differentiable pooling, MinCUT pooling, etc.), such a deterministic pooling operator is considerably more computationally efficient and thus scalable to larger stock graphs. Experimentations on the S&P500 stock index demonstrate that the StockPool operator outperforms existing graph pooling strategies on the prediction of price movements. Finally, different graph pooling methods are utilized to create a set of highly uncorrelated GNN models; these are used to construct a graph ensemble model with an improved performance.

1. INTRODUCTION

In the past decade, there has been a surge in the development of graph neural networks (GNN) for applications on irregular domains; these vary from social networks [1], traffic prediction [2, 3], through to gene inference [4]. The complex and hierarchical nature of the stock market represents therefore an ideal domain for the operation of GNNs on an irregular set of heavily structured stock graphs. However, an issue with current GNNs is that the propagation of information is limited to only between individual edges in a graph, making it difficult to learn hierarchical relationships between the nodes (e.g. stocks) in the graph. This limitation may be bypassed through graph coarsening (or pooling), which allows for a high-level representation of the stock graph at each layer of a GNN model to be obtained. This can be achieved using a subset of graph pooling operators, known as hierarchical pooling [5].

At present, there exists a variety of hierarchical pooling operators which employ different methodologies and auxiliary objectives, including Top-K pooling [6], SAG (Self-Attention Graph) pooling [7], ASA (Adaptive Structure Aware) pooling [8], Differentiable pooling [5], MinCUT pooling [9], and the more recent Memory-based Pooling [10]. The work in [11] generalizes pooling operations under a common framework referred to as the SRC (Select, Reduce, and Connect).

The SRC framework considers the graph pooling operation as a combination of three constituent operations: *Select*, *Reduce*, and *Connect*, whereby different pooling operations employ a combination of these operations in different ways. The *Select* operation is responsible for obtaining the new supernodes of the coarsened (or pooled) graph, the *Reduce* operation obtains the new embeddings of each supernode, and finally the *Connect* operation determines the edge connections between the supernodes of the coarsened graph. A further useful categorization of graph pooling operations is regarding whether or not they are trainable. Trainable pooling operators (e.g. DiffPool [5], MinCUTPool [9], SAGPool [7], MemPool [10]) are able to learn the optimal pooling strategy in an end-to-end fashion against an objective function, while non-trainable operators (e.g. Graclus [12], NDP [13], LaPool [14]) typically use pre-defined algorithms. As suggested in [11], non-trainable methods are particularly useful when there is strong prior information regarding the most desired pooling behaviour (e.g. maintaining graph connectivity, homogeneity of node embeddings, etc.). Trainable operators, on the other hand, require fewer assumptions about the desired pooling behaviour. In the case of a stock graph, due to its unpredictable and time-varying nature, a trainable pooling operator may prove advantageous because of its ability to learn rather than assume a desired pooling behaviour [15, 16].

Note that trainable operators may also benefit from prior assumptions regarding the structure of the graph [16]. This is because the end-to-end optimization of the graph pooling strategy is non-convex [5] and thus many trainable pooling methods introduce auxiliary objective loss functions to obtain an improved convergence performance. The highly influential differential pooling operator [5], for example, uses the link prediction loss to encode the intuition that nearby nodes should be clustered together, whereas MinCUT pooling [9] introduces an auxiliary loss function to minimize the so-called mincut loss. However, while these prior assumptions have proven effective for most graph structures (e.g. social networks, road traffic networks, document citation net-

works), they may not perform particularly well when applied to the stock graph problem. This is due to the inherently non-branching structure of many unweighted stock graphs, i.e. the maximum edge-distance between any two nodes is generally short. This violates many prior graph assumptions which are inherent to existing pooling methods. For example, the link prediction loss utilizes the assumption that nearby nodes should be grouped together; however, in a stock graph, edge-distance between nodes are relatively short, thus this additional auxiliary loss is not likely to improve the overall convergence result.

To this end, we introduce the *StockPool* graph pooling operator, which utilizes domain-aware prior assumptions which are specifically tailored for the stock graph. Comprehensive analysis and simulation case studies conclusively demonstrate the computational and performance advantages of the proposed pooling operator against other pooling operators such as DiffPool and MinCUTPool.

2. PRELIMINARIES

2.1. Graph Representation of the Stock Market

A graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ with n nodes consists of a set of nodes $\mathcal{N} = \{x_1, x_2, ..., x_n\}$ and a set of edges $\mathcal{E} = \{e_{ij}, e_{ik}, ...\}$, where $e_{ij} = 1$ indicates a connection between nodes i and j, whilst $e_{ij} = 0$ corresponds to an absence of an edge. Each node carries an m-dimensional embedding, $x_i \in \mathbb{R}^m \quad \forall i \in 1, 2, ..., n$ (i.e. vector representation of node information). Further, edge connections in a graph can be represented through an adjacency matrix, $\boldsymbol{A}$, whose corresponding (i, j)-th entry encodes the presence or absence of a connection between nodes i and j.

In the case of a stock graph, nodes can represent individual stock information with the edges as the corresponding relative distances between the stocks. We employ the dynamic time warping (DTW) distance [17, 18] as a metric to measure relative distances between the individual stocks. Denote by $\mathcal{W} = \{w_1, w_2, ..., w_K\}$ the warp path between two stock price sequences, $\mathcal{X}$ and $\mathcal{Y}$. The warp distance is defined as

$$Dist(\mathcal{W}) = \sum_{k=1}^{K} Dist(w_{ki}, w_{kj})$$

whilst the DTW distance between the two price sequences $\mathcal{X}$ and $\mathcal{Y}$ corresponds to the minimum warp distance, given by

$$DTW(\mathcal{X}, \mathcal{Y}) = \min_{\mathcal{W}} Dist(\mathcal{W}) = \min_{\mathcal{W}} \sum_{k=1}^{K} Dist(w_{ki}, w_{kj})$$

Finally, the DTW edge-construction constraint is defined as

$$e_{ij} = \begin{cases} 1, & \text{if } DTW_{ij} < \epsilon \\ 0, & \text{otherwise} \end{cases} \quad \forall i, j = 1 : n,\ i \neq j$$

The problem of stock forecasting is a time series problem, and thus the stock graph must encapsulate a time series element. For simplicity, we employ the log-returns of the past N trading day as the node embedding. In particular, $N = 10$ days is shown to attain the best overall performance amongst the considered GNN models, while log-returns are used due to their approximate Gaussian property [19].

2.2. First-Order Approximation of Graph Convolution

The first-order approximation of spectral convolution on graphs forms the basis of graph convolutional networks (GCN) [20]. Consider a graph signal, $\boldsymbol{x} \in \mathbb{R}^n$, (i.e. a scalar for each node) with a graph convolutional filter, $\boldsymbol{h}$. A graph convolution operation of order $(M-1)$ can then be described through the graph shift operator (GSO) $\boldsymbol{S}$ [16], as

$$\boldsymbol{y} = \boldsymbol{h} \circledast \boldsymbol{x} = h(0)\boldsymbol{x} + h(1)\boldsymbol{S}\boldsymbol{x} + \cdots + h(M-1)\boldsymbol{S}^{M-1}\boldsymbol{x}$$

where $\circledast$ denotes convolution. The first-order approximation of graph convolution ($M = 2$) is then given by

$$\boldsymbol{y} = \boldsymbol{h} \circledast \boldsymbol{x} = h(0)\boldsymbol{x} + h(1)\boldsymbol{S}\boldsymbol{x} \tag{1}$$

A common GSO is the normalized adjacency matrix, although isometric GSOs may also be used due to their desirable boundedness properties [21]. Given $\boldsymbol{L}_n \in \mathbb{R}^{n \times n}$ as the normalized graph Laplacian and $\boldsymbol{D} \in \mathbb{R}^{n \times n}$ as the diagonal degree matrix of the graph, the GSO (in our case the normalized adjacency matrix) can be defined as

$$\boldsymbol{S} = (\boldsymbol{L}_n - \boldsymbol{I}_n) = \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}} \tag{2}$$

Upon substituting (2) into (1), we arrive at

$$\boldsymbol{y} = h(0)\boldsymbol{x} + h(1)\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{x}$$

A further simplification, with $h(0) = h(1) = \theta$, yields [22]

$$\boldsymbol{y} = \theta(\boldsymbol{I}_n + \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}})\boldsymbol{x}$$

However, such simplification may cause over-reduction of the parameter space leading to a decrease in performance [16].

Generally, k convolutional filters can be used, while instead of the graph signal, $\boldsymbol{x} \in \mathbb{R}^n$, it is possible to employ a feature embedding matrix, $\boldsymbol{X} \in \mathbb{R}^{n \times m}$ (i.e. each node carries an m-dimensional signal). This modification yields

$$\boldsymbol{Y} = (\boldsymbol{I}_n + \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}})\boldsymbol{X}\boldsymbol{\Theta} \tag{3}$$

where $\boldsymbol{\Theta} \in \mathbb{R}^{m \times k}$ represents the set of trainable convolution filter weights, and $\boldsymbol{Y} \in \mathbb{R}^{n \times k}$ the output feature embedding matrix of the convolution operation.

To prevent instability during the graph convolution process, [22] further proposes a modification in the form of a normalization of (3), based on

$$\hat{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}_n,\ \hat{\boldsymbol{D}}_{ii} = \sum_{j=0} \hat{\boldsymbol{A}}_{ij}$$

to yield

$$\boldsymbol{Y} = \hat{\boldsymbol{D}}^{-1/2}\hat{\boldsymbol{A}}\hat{\boldsymbol{D}}^{-1/2}\boldsymbol{X}\boldsymbol{\Theta}$$

2.3. Graph Pooling and the SRC Framework

Graph pooling can be employed in conjunction with GCN layers to extract high-level features within the graph structure [16]. Denote by P_l, $\boldsymbol{H}^{(l)}$ and $\boldsymbol{\Theta}_l$, the pooling operator, feature embedding matrix, and the trainable convolutional filter weights at a layer l of the GNN, respectively. Further, denote by σ a non-linear activation function. The single-layer update of a GNN with a convolution and pooling layer is then given by

$$\boldsymbol{H}^{(l+1)} = P_l\left\{\sigma\left(\hat{\boldsymbol{D}}^{-1/2}\hat{\boldsymbol{A}}\hat{\boldsymbol{D}}^{-1/2}\boldsymbol{H}^{(l)}\boldsymbol{\Theta_l}\right)\right\}$$

Given that the pooling operation performs effectively a downsampling of the original graph, we have $\boldsymbol{H}^{(l)} \in \mathbb{R}^{n\times m}$ and $\boldsymbol{H}^{(l+1)} \in \mathbb{R}^{n'\times m'}$, where $n' < n$ (i.e. number of nodes of the coarsened graph is reduced). Different pooling operations can be described using the SRC framework [11]. With $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ and $\mathcal{G}' = (\mathcal{N}', \mathcal{E}')$ as the original and coarsened graph respectively, the pooling operation can be denoted by $\mathcal{G}' = POOL(\mathcal{G})$. Recall that the SRC framework formalizes graph pooling as a sequence of three distinct steps, the select (SEL), reduce (RED), and connect (CON) steps. For graphs $\mathcal{G}$ and $\mathcal{G}'$, with the respective number of nodes n and n', the SRC framework can be described as

$$S = \{S_k\}_{k=1:n'} = SEL(\mathcal{G})$$

$$X' = \{X'_k\}_{k=1:n'} = \{RED(\mathcal{G}, S_k)\}_{k=1:n'}$$

$$\mathcal{E}' = \{\mathcal{E}'_{i,k}\}_{i,k=1:n'} = \{CON(\mathcal{G}, S_i, S_k)\}_{i,k=1:n'}$$

The symbol S denotes the set of supernodes of the graph $\mathcal{G}'$, X' stands for the set of node embeddings for each supernode, and $\mathcal{E}'$ designates the edge connection between each pair of supernodes. It is important to note that different pooling operations approach the SEL, RED, and CON steps differently.

2.4. GNN Architecture with Graph Pooling

The architecture proposed in this work involves a sequence of GCN and hierarchical pooling layers, whereby each layer progressively captures more and more high-level features of the stock market, based on the graph coarsening process. The output embedding of the architecture is fed into a multi-layer perceptron (MLP) to perform multi-stock prediction. The weights of the GCN and graph pooling layers can be trained in an end-to-end fashion with binary cross-entropy (BCE) as a loss function. Denote by $y_v(t)$ the ground-truth binary price shift for a stock v at time t, and by $z_v(t)$ the sigmoid-bounded output of the model. The corresponding BCE loss across the time interval T can then be computed as

$$L_{bce} = -\frac{1}{nT}\sum_{t}^{T}\sum_{v\in N} y_v(t)log(z_v(t)) + (1-y_v(t))log(1-z_v(t))$$

3. THE STOCK POOLING OPERATOR

The non-branching nature of stock graphs makes many existing assumptions, that are routinely incorporated into trainable pooling operations (through link prediction loss, mincut loss, etc.), inadequate. To this end, we propose a finance-specific pooling operation, referred to as the *StockPool*, which incorporates the domain knowledge specific to the stock market to achieve an improved end-to-end performance. In particular, StockPool leverages the assumption that stocks within the same industries and sub-industries share similar underlying economic features, thereby allowing for the formation of natural clustering between such stocks. This implies that the cluster assignment of the StockPool operation is deterministic but domain-aware, thus reducing the overall computational cost.

Consider a graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ with n stocks, n_{sub} sub-industries, and n_{main} main industries. The StockPool operation represents a two-step pooling procedure which comprises two hard-cluster assignment (or downsampling) matrices $\boldsymbol{S_1} \in \mathbb{R}^{n\times n_{sub}}$ and $\boldsymbol{S_2} \in \mathbb{R}^{n_{sub}\times n_{main}}$. Denote by $sub(i)$ a many-to-one mapping of a stock i to its corresponding sub-industry, and by $ind(i')$ a similar mapping of a sub-industry i' to its corresponding main industry. The hard-cluster assignment matrices can then be defined as

$$\boldsymbol{S_1}_{i,j} = \begin{cases} 1, & \text{if } sub(i) = j \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

$$\boldsymbol{S_2}_{i,j} = \begin{cases} 1, & \text{if } ind(i) = j \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

Figure 1 illustrates the deterministic clustering assignment of the StockPool operation, which clusters individual stock nodes to form the sub-industry graph, prior to performing a final clustering of sub-industry nodes to form the sector graph.

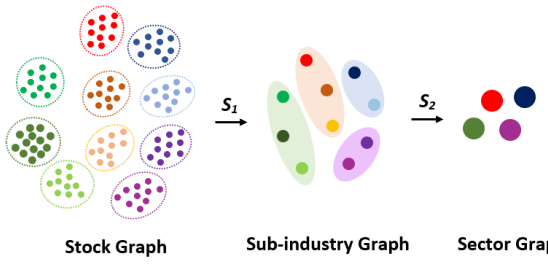

Fig. 1. Illustration of Two-Layer hierarchical *StockPool*.

Upon employing the hard-clustering assignment matrices in (4) and (5), the corresponding reduced feature embedding matrix, $\boldsymbol{X'}$, can be computed through a sub-GNN model as

$$\boldsymbol{X'} = \boldsymbol{S}^{\top} \cdot \text{GNN}(\boldsymbol{A}, \boldsymbol{X})$$

The reduced adjacency matrix, $\boldsymbol{A'}$, of the coarsened graph can be similarly obtained from the hard-clustering assignment matrices, to yield

$$\boldsymbol{A'} = \boldsymbol{S}^\top \boldsymbol{A}\boldsymbol{S}$$

4. GRAPH ENSEMBLE LEARNING

Ensemble methods employ a set of individually trained weak learners whose predictions are combined to obtain improved generalization performance [23, 24, 25]. While several methods have been proposed to incorporate ensemble learning into graph models [26, 27], the area is still under development. The simulations on the stock market suggest that GNN models with different graph pooling operators tend to exhibit a degree of uncorrelatedness. This desirable property motivates us to exploit a graph ensemble strategy, whereby GNNs with different graph pooling operators serve as a basis for a graph ensemble model. In particular, the StockPool, DiffPool, and MinCUTPool operators were employed to form the graph ensemble, as shown in Figure 2. Denote by $DP_i(t)$, $MCP_i(t)$ and $SP_i(t)$, the output prediction of the DiffPool, MinCUTPool, and StockPool GNNs, respectively, for a stock i at a time t. The output of the ensemble $Ens_i(t)$ is then given by

$$Ens_i(t) = \frac{DP_i(t) + MCP_i(t) + SP_i(t)}{3}$$

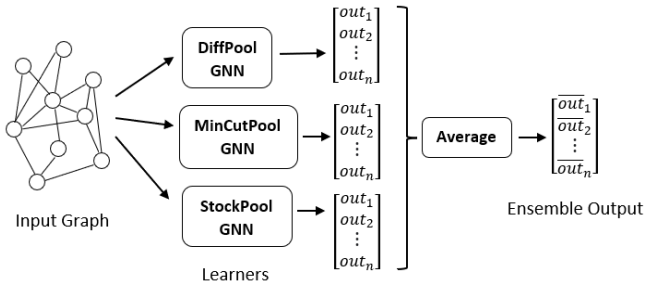

Fig. 2. A graph ensemble architecture, comprising DiffPool, MinCUTPool, and StockPool graph neural networks (GNN).

5. SIMULATIONS

Performance of the individual and ensemble GNN models was investigated on 457 stocks within the S&P500 index between 2015-01-01 and 2022-01-30. The dataset consists of 11 GICS (Global Industry Classification Standard) industries, which was partitioned into the train and test sets with an 80-20 train-test split; this gave 663,294 training samples and 165,710 test samples. Evaluation of the models was performed using the binary accuracy and the Sharpe ratio; the mean and standard deviation was calculated over 10 experiments for each model. The Sharpe ratio was computed through a backtesting simulation on the test set, assuming an unweighted portfolio of all 457 stocks. Further, the performance of each model was benchmarked against the base S&P500 market movement (i.e. by taking a long position on the unweighted portfolio), as shown in Table 1. Overall, the ensemble model attained the best mean performance in terms of both test accuracy and the Sharpe ratio - in addition to having a reduced variance relative to its constituent sub-models. Figure 3 compares the backtesting simulation of the three top performing models, against the ensemble model.

Model	Test Accuracy(%)	Sharpe Ratio
S&P500 Market (Long)	52.18	1.2613
SAGPool	52.06 (0.0010)	0.9569 (0.16)
TopKPool	52.15 (0.0002)	1.1240 (0.13)
MinCutPool	52.21 (0.0022)	1.3363 (0.23)
DiffPool	52.37 (0.0044)	1.6160 (0.44)
StockPool	**52.57 (0.0021)**	**1.7222 (0.63)**
Ensemble	**52.61 (0.0021)**	**1.7306 (0.27)**

Table 1. Test accuracy and Sharpe ratio, evaluated as Mean (STD), of the GNN models and the base S&P500 market.

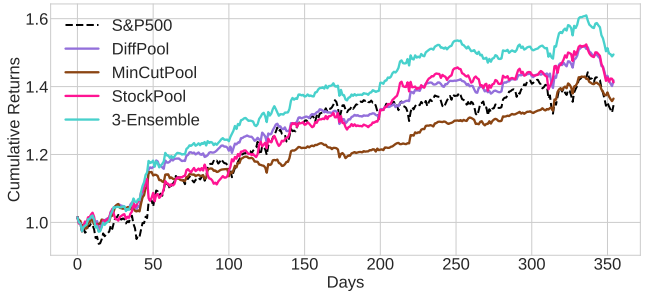

Fig. 3. Evaluation of GNN models against S&P500 market.

6. CONCLUSIONS

We have introduced a finance-specific graph pooling operator, referred as the *StockPool*, with the aim to improve computational efficiency while maintaining enhanced end-to-end performance, compared to existing pooling operations. This has been achieved by incorporating domain knowledge related to the stock graph into the graph coarsening process. Such a domain-aware graph pooling operator has been developed by clustering stocks into their corresponding industries, under the assumption that such stocks share key underlying economic features. We have also investigated the use of different graph pooling operators as a basis for constructing highly uncorrelated GNN models, a pre-requisite for a successful ensemble model. Experimentation based on the S&P500 stock index have demonstrated that the graph ensemble model outperforms its constituent sub-models, as well as the general movement of the market.

7. REFERENCES

[1] T. Bai, Y. Zhang, B. Wu, and J.-Y. Nie, "Temporal graph neural networks for social recommendation," in *Proceedings of the IEEE International Conference on Big Data (Big Data)*, 2020, pp. 898–903.

[2] Z. Lu, W. Lv, Z. Xie, B. Du, and R. Huang, "Leveraging graph neural network with LSTM for traffic speed prediction," in *Proceedings of the IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation*, 2019, pp. 74–81.

[3] B. S. Dees, Y. L. Xu, A. G. Constantinides, and D. P. Mandic, "Graph theory for metro traffic modelling," in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021, pp. 1–5.

[4] Y. Yuan and Z. Bar-Joseph, "GCNG: Graph convolutional networks for inferring gene interaction from spatial transcriptomics data," *Genome Biology*, vol. 21, 12 2020.

[5] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018.

[6] H. Gao and S. Ji, "Graph U-nets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, pp. 4948–4960, 2022.

[7] B. Knyazev, G. W. Taylor, and M. Amer, "Understanding attention and generalization in graph neural networks," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019.

[8] E. Ranjan, S. Sanyal, and P. Talukdar, "ASAP: Adaptive structure aware pooling for learning hierarchical graph representations," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 5470–5477, 04 2020.

[9] F. M. Bianchi, D. Grattarola, and C. Alippi, "Spectral clustering with graph neural networks for graph pooling," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 874–883.

[10] A. H. Khasahmadi, K. Hassani, P. Moradi, L. Lee, and Q. Morris, "Memory-based graph networks," 2020. [Online]. Available: https://arxiv.org/abs/2002.09518

[11] D. Grattarola, D. Zambon, F. M. Bianchi, and C. Alippi, "Understanding pooling in graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–11, 2022.

[12] I. S. Dhillon, Y. Guan, and B. Kulis, "Weighted graph cuts without eigenvectors a multilevel approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 11, pp. 1944–1957, 2007.

[13] F. M. Bianchi, D. Grattarola, L. Livi, and C. Alippi, "Hierarchical representation learning in graph neural networks with node decimation pooling," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 5, pp. 2195–2207, 2022.

[14] E. Noutahi, D. Beaini, J. Horwood, S. Giguère, and P. Tossou, "Towards interpretable sparse graph representation learning with laplacian pooling," 2019. [Online]. Available: https://arxiv.org/abs/1905.11577

[15] L. Stankovic, D. Mandic, M. Dakovic, M. Brajovic, B. Scalzo, S. Li, and A. G. Constantinides, "Graph signal processing. Part III: Machine learning on graphs, from graph topology to applications," 2020. [Online]. Available: https://arxiv.org/abs/2001.00426

[16] L. Stankovic and D. Mandic, "Understanding the basis of graph convolutional neural networks via an intuitive matched filtering approach," 2021. [Online]. Available: https://arxiv.org/abs/2108.10751

[17] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, ser. AAAIWS'94. AAAI Press, 1994, p. 359–370.

[18] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intelligent Data Analysis*, vol. 11, no. 5, p. 561–580, oct 2007.

[19] P. Mota, "Normality assumption for the log-return of the stock prices," *Discussiones Mathematicae. Probability and Statistics*, vol. 32, 01 2012.

[20] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[21] B. S. Dees, L. Stankovic, M. Dakovic, A. G. Constantinides, and D. P. Mandic, "A class of doubly stochastic shift operators for random graph signals and their boundedness," 2019. [Online]. Available: https://arxiv.org/abs/1908.01596

[22] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *Proceedings of the 5th International Conference on Learning Representations*, ser. ICLR '17, 2017.

[23] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *Journal of Artificial Intelligence Research*, vol. 11, pp. 169–198, aug 1999.

[24] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, Aug 1996.

[25] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct 2001.

[26] P. Goyal, S. Raja, D. Huang, S. R. Chhetri, A. Canedo, A. Mondal, J. Shree, and C. Jawahar, "Graph representation ensemble learning," in *2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 2020, pp. 24–31.

[27] X. Hou, P. Qi, G. Wang, R. Ying, J. Huang, X. He, and B. Zhou, "Graph ensemble learning over multiple dependency trees for aspect-level sentiment classification," 2021. [Online]. Available: https://arxiv.org/abs/2103.11794