CancelOut GCN Diffusion(CoutGCN): Finding the Influential Spreaders to Diffuse Information

Koyena Chowdhury
Computer Science and Engineering
National Institute of Technology
Durgapur, India
koyenachowdhury02@gmail.com

Amit Paul
Computer Science and Engineering
National Institute of Technology
Durgapur, India
ap.18cs1501@phd.nitdgp.ac.in

Animesh Dutta
Computer Science and Engineering
National Institute of Technology
Durgapur, India
animesh@cse.nitdgp.ac.in

2023 15th International Conference on COMmunication Systems & NETworkS (COMSNETS) | 978-1-6654-7706-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/COMSNETS56262.2023.10041335

***Abstract*—Influential spreaders are the most efficient nodes that can communicate with a vast number of users in a network in the easiest and fastest way. Information spreading, viral social marketing, immunization and controlling the spread of rumors, diseases or viruses are some of the top-notch applications for which identifying the source nodes has become a pivotal job. The computational complexities of the state-of-the-art approaches make it challenging to locate these critical nodes. To minimize such loopholes in the existing methods, in this paper we propose a method called CancelOut GCN Diffusion (CoutGCN) to detect the influential spreaders and their effect on diffusion prowess, which is a much simpler process than the previous proposals. Here, we merge the concepts of Graph Convolutional Network (GCN), feature selection, clustering and community detection algorithms to locate the influential nodes to diffuse information throughout the network. The preprocessing layer CancelOut is a feature selection technology, used to determine the node rankings, while GCN is used for node classification. Different clustering algorithms, particularly Kmeans and Kmedoids, and community detection techniques, viz. LGA and Louvain, are implemented to originate the labels for the training of GCN class prediction. Finally we compared all these algorithms with the previous four benchmark methods, such as Adaptive Degreerank, Voterank, K-shell and EnRenew on three real-world social networks to show how and where each method will give the best result according to the application. Experimentally we found that community detection algorithms worked better when fewer initial spreaders were present, as they would be located in more scattered parts of the network. Clustering models outperformed in a denser network when more influential spreaders were chosen.**

***Index Terms*—influential spreaders, CoutGCN, information diffusion, node ranking, node labeling, clustering and community detection**

I. INTRODUCTION

Social media networks have become vast resources of information and influence. In the modern era, these complex networks are the most effective tools to accelerate the pace of information. Social media has given a platform where any user can have their views and opinions which can be delivered to anyone, irrespective of the geographical distance among users. Some information spread fast due to its informative content, looks, feel or interest. Another aspect of information spreading is the influence of the user who has released the content. In social media, there are many such users who can influence a vast majority of other users faster than others. In complex network jargon, they are called influential spreaders [1].

For example, if a company or an individual wants to advertise its product through YouTube, it will try to find channels with the highest subscribers and views. The channels are the influential spreaders in this case. Another leading example could be when actors want to make their films viral through trailers and promote them through top-notch celebrities on online social media platforms such as Instagram or Facebook. Thus, finding the most influential spreaders in a social network is an important and challenging task. Moreover, finding influential nodes in a graph to maximize or control the propagation of information flow is a significant problem if we want to spread information easily and effortlessly [2].

In this paper, we propose an innovative approach to this problem by combining a few research areas like deep convolutional networks, feature selection, information diffusion, clustering and community detection. Our contributions are summed up as follows:

- We propose a model that produces better results when compared with other existing state-of-the-art methods at a lesser time complexity, i.e., $O(|E|)$.
- We observe that other than clustering algorithms, community detection algorithms can also be used for node labeling to train the GCN loss function in order to find the seed nodes that can aptly diffuse information faster in some cases.
- We compare different clustering and community detection algorithms and observe their applicability according to the infection rate, number of initial spreaders and type of the network.

II. STATE OF THE ART

Throughout the last decade or so, numerous methods have been devised with significant contributions to identify influential nodes diffusion and maximizing or controlling the spreading process in the network. Traditional approaches to find high-ranked hub nodes are: k-shell decomposition [3], mixed degree decomposition [4], weighted kshell degree neighborhood method [5], improved k-shell decomposition [6], VoteRank++ [7], collective influence method [8], semi-local centrality [9], global structure based centrality [10], semi-global triangular centrality [11]. But most of these methods suffer from high time and computational complexity [12].

Some greedy algorithms also have been proposed as described in the papers [13] and [14], however they are computationally very costly.

A few deep learning based methods have also been proposed in this field. Atwood et al. [15] introduced the diffusion convolutional neural model for graph data using the Markov process for information diffusion. Topo-LSTM [16], on the other hand, explored the cascade structure on dynamic Directed Acyclic Graphs (DAG) using representation learning on graphs. Recently, Cao et al. designed a mechanism on graph structure and temporal features, namely Dynamic Structural Temporal GNN (DySTGNN) [17] to overcome the limitation of the impression of changing preferences of the users eventually. Rumor spreaders on Twitter were identified using weak supervised learning [18] by implementing GCN. Ye et al. stated an end-to-end computational model based on GNN [19] using Graph Attention Network (GAT) on the Sina Weibo network to find the vital users. CGNN [20] was proposed by merging two deep learning methods CNN and GNN, to fix the issue. GCNFusion [21] combined feature selection and GCN to locate the important nodes to diffuse the information. As real-world networks are mostly unlabeled, we propose a more generalized version of GCNFusion in unsupervised way with a few modifications. To investigate further the spreading dynamics of influential nodes, we use some community detection and clustering algorithms.

III. PROBLEM FORMULATION

A complex network can be represented as a graph in which vertices are the users and edges are the relationships between a pair of users. Given a graph $G = \langle V, E \rangle$, where n is the number of nodes i.e. $|V| = n$, where $V = \{v_1, v_2, ...v_n\}$, and $|E|$ is the connections between them such that $E = \{e_1, e_2, ...e_m\}$. Here $A = a_{ij} \in V \times V$, where A is the adjacency matrix and $a_{ij} = 1$ denotes that there exists an edge between node 'i' and 'j' whereas $a_{ij} = 0$ suggests no edge existing between them. We need to find the set of nodes $S = \{s_1, s_2, s_3, ..., s_k\}$ where $S \subset V$ and k is the number of influential nodes that take part in the spreading process.

IV. METHODOLOGY

In this section, we describe how we have devised a deep learning based method to identify influential nodes in a network that can propagate information to each node of a social network most effectively and efficiently. Here, we apply the CancelOut method [22], which is a part of data pre-processing, on networks to provide the ranking of all nodes. Separately, we apply the clustering technique to generate the node labels. The node ranking generated from CancelOut layer and node labels from the clustering and community detection techniques are taken as input of GCN [23]. GCN is trained using the loss function to generate highly influential nodes in the network. Fig. 1 illustrates the flowchart of our model CoutGCN, which is described more precisely in the following sections.

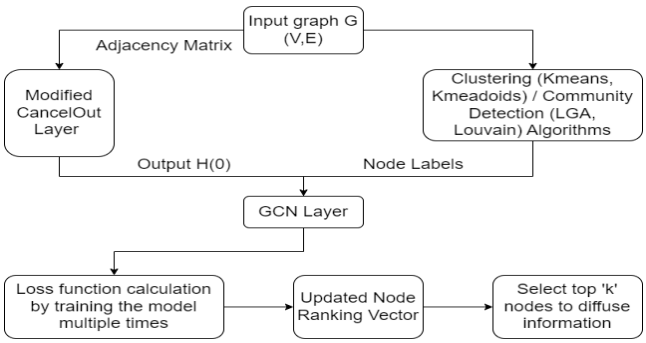


Fig. 1: Flow Chart of the Proposed Model

A. Generation of Node Ranking

Feature selection is a method used to reduce dimensionality and complexity while training the model on only relevant and essential features in the most efficient way. CancelOut, which is a feature selection and ranking technique, takes a matrix X as an input vector, W^{CO} as weight vector and σ as an activation function. We modified the CancelOut method for node ranking instead of feature ranking. We argue that if the method can generate the ranking for features, it can also be used for node ranking. We took each node as a vector with information about its neighbors and non-neighbors. The modified CancelOut method is as follows: $H^{(0)} = \tilde{A} \cdot \sigma(W^{CO})$ Here $\tilde{A}$ Denotes the adjacency matrix of the graph, σ is the activation function (we use the Sigmoid function), W^{CO} is basically a trainable weight matrix used for finding the node rankings, and $\cdot$ implies element-wise multiplication. This layer outputs positive and negative values for each node that is ranked. This output of the CancelOut layer is used as the feature embedding of the GCN layer where the value of W_i^{CO}, where $[i \in \{1, 2, ..., n\}]$ becomes 0 if it is a negative number, i.e. this node does not have any effect in the diffusing process and the higher positive values signify more influential nodes.

B. Clustering and Community Detection

Most of the real-world networks are devoid of ground truth. To fill the gap, we sometimes use unsupervised techniques to learn the labels of nodes in the network. In this paper, we use the K-Means, K-Medoids and two other community detection techniques like Local Group Assimilation (LGA) [24], and Louvain [25], which provide more insight into the workings.

C. Modeling with GCN

GCN is a type of Graph Neural Networks (GNNs) [26], which belongs to a class of deep learning approaches that can be directly applied to networks for edge level, node level and graph level predictions. A short description of the working principle to produce node level predictions of GCN is stated as follows. The structural and feature-based information of the nodes is fed through message passing layers to produce node embeddings. A node in the graph aggregates the details of its adjacent nodes and updates its own information to generate the

updated node embedding. Mathematically, the next layer embedding of a GCN layer is, $H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{l}W^{l})$ where $\tilde{D}_{ij} = \sum_j \tilde{A}_{ij}$ is the diagonal degree matrix, $\tilde{A} = A + I_N$, where A is the adjacency matrix, I is the identity matrix of size n, H^l is the previous layer embedding and W^l implies trainable weight vector. Here σ is the ReLU activation function, which will produce the same output for positive values but will make it 0 in the case of negative numbers. Stochastic Gradient Descent(SGD) is used to minimize the loss while training. A node belongs to the class with the maximum value after applying the Softmax function. Cross Entropy Loss (CEL) is used to update the weight parameters such that,
$\mathcal{L} = -\sum_{f=1}^{F} Y_f ln Z_f + \lambda_1 var(\frac{W^C O}{N}) + \lambda_2 ||\frac{W^C O}{N}||$,
where f is the number of class, Y is the actual output, and Z_f is our predicted output [22].

D. Benchmark simulator

We used Susceptible–Infected–Recovered (SIR) epidemic simulator [27] to disseminate the information in the graph and analyse the spreading capability of the proposed model. In the beginning, all the nodes are in a susceptible state except the nodes that are infected, as they are the "influential originators". These spreaders randomly infect their neighbor nodes with probability μ, which is slightly higher than the threshold probability μ_c. The nodes go into the recovery state with probability β and are removed from the graph for the diffusion process as indicated by [28]. μ of a network is calculated by $\mu = \frac{\langle k^2 \rangle}{(\langle k^2 \rangle - \langle k \rangle)}$, where $\langle k \rangle$ is the average degree and $\langle k^2 \rangle$is the second order average degree of each node in the graph. The infection rate λ is defined as $\lambda = \frac{\mu}{\beta}$ which plays a key role in the spreading process. The process of spreading stops when all the nodes are infected.

V. EXPERIMENTAL SETUP

A. Datasets Description

We used three real-world network datasets to evaluate the functionality of our proposed model. These undirected and unweighted networks are described as follows. (1) Jazz [29] is a social network of jazz musicians and relationships between them indicate the collaboration between the musicians. (2) Caenorhabditis Elegans [30] is the metabolic network of C. Elegans, which is a type of worm, where nodes are the proteins and edges are the interaction between them. (3) Hamster [31] is a social relationship network of hamsterster.com where the nodes are the users of the website.

TABLE I: Dataset Description [$|V|$ represents number of nodes, $|E|$ indicates number of edges, $\langle k \rangle$ stands for network average degree, $\langle c \rangle$ symbolises average clustering co-efficient, λ denotes the infection rate, and α is the spectral norm.]

Network	$\|V\|$	$\|E\|$	$\langle k \rangle$	$\langle c \rangle$	λ	$\frac{\|E\|}{\|V\|}$	α
Jazz	198	2,742	27.69	0.61	0.045	13.848	40.02
C. Elegans	453	2,025	8.94	0.64	0.166	4.47	26.3
Hamster	2,426	16,631	13.7	0.53	0.057	6.855	50.02

B. Performance Metrics

We use two metrics, (1) Final affected scale and (2) Average shortest path length, that are pointed out in the research paper [1]. These metrics are defined as follows:

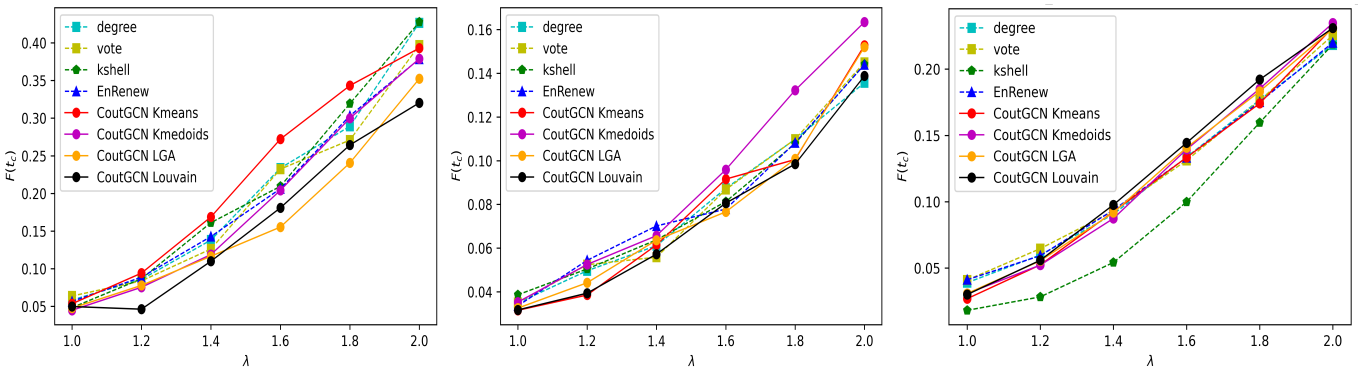
1) **Final affected scale $F(t_c)$:**
The final diffusion capability of the first influential spreaders of a graph $F(t_c)$ is defined as
$$F(t_c) = \frac{n_{R(t_c)}}{n}$$
where $n_{R(t_c)}$ is the number of removed nodes at time step t_c when the stable state is reached in the SIR model.
2) **Average shortest path length L_s:**
The average shortest path length L_s is evaluated taking into consideration the structural properties of a network such that
$$L_s = \frac{1}{|S|(|S|-1)} \sum_{u,v \in S, u \neq v} l_{u,v}$$
where S denotes a pair of initial spreaders and $l_{u,v}$ is the shortest path length between the nodes u and v. The higher value of L_s indicates how distant the spreaders are from each other. Hence better diffusion process will be performed if L_s is high.

C. Computational Complexity

We used only 2 GCN layers as it is observed that friendly nodes in social networks share numerous common neighbors mostly within 2 hops [32]. Multiple hops increase the chance of noise propagation, thus leading to the over-smoothening problem. GCN has an overall time complexity of $O(LEd + Lnd^2)$, where L represents the number of layers, d denotes the dimension of node embedding vectors; here node adjacency vectors and E, n signifies the edges and nodes of the graph, respectively. A few approximations and simplifications can reduce this complexity to $O(|E|)$ [33]. Similarly, The space complexity of GCN is $O(Lnd + Ld^2)$ [23]. The time complexity of our modified CancelOut layer is $O(n)$ [22]. The Kmeans algorithm time complexity is $O(n^2)$. The K-Medoids is known to have a time complexity of $O(k(n-k)^2)$. The complexity of LGA is $O(n^2)$ [24], and that of Louvain is $O(n \log(n))$ [34]. So, computationally, we see that the K-Medoids algorithm takes less time than Kmeans. In the case of community detection algorithms, Louvain performs better than LGA if time is a concern. CoutGCN is the combined version of all these methodologies, having the best time complexity as $O(n + E + n^2)$ for clustering algorithms and $O(n + E + n \log(n))$ for community detection, which can be compared with $O(|E|)$ if the graph is sparse.

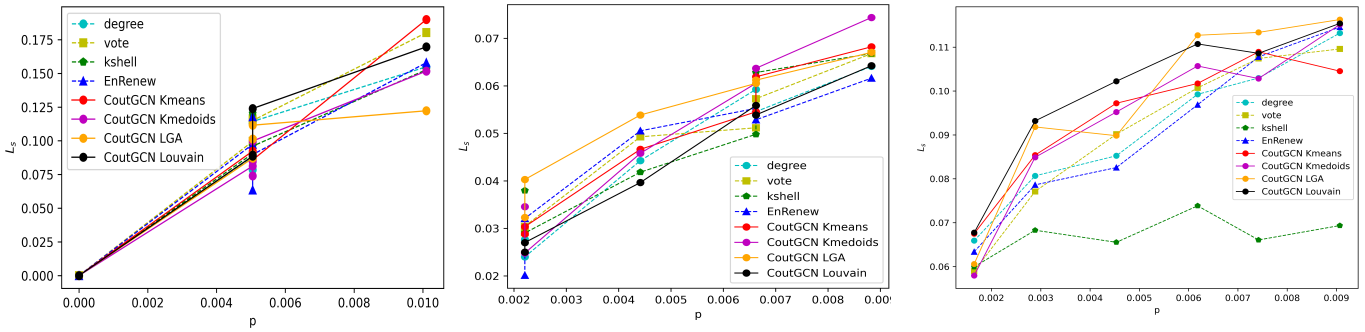
VI. RESULT ANALYSIS

We perform two experiments for each of the three real-world datasets to examine the efficiency of our algorithm. Experiment 1 plots the outcomes of the affected scale concerning different infected rates of important nodes, whereas Experiment 2 examines the average shortest path lengths for



(a) Jazz (b) C.Elegans (c) Hamster

Fig. 2: Experiment 1: Infection Rate λ v/s Final Affected Scale $F(t_c)$



(a) Jazz (b) C.Elegans (c) Hamster

Fig. 3: Experiment 2: Average shortest path lengths (L_s) v/s Disparate Segments of Initiators (p)

different ratios of initial spreaders. We compared our model with four other existing methods, (1) Adaptive Degreerank [13], (2) Voterank [1], (3) K-shell [5] and (4) EnRenew [35].

Experiment 1: The impact of the infection rate (λ) on the affected scale $F(t_c)$ is calculated to compare the results of different methods. As illustrated in Fig: 2, (λ) value is kept between 1 and 2 $(1 \leq \lambda \leq 2)$ so that we can examine the behavior of infected nodes for both small and large values while keeping the spreading acceleration speed moderate. For a denser network like Jazz, where the average number of edges $(\frac{|E|}{|V|})$ per degree is higher, Kmeans has worked well for most of the values of λ. Whereas for more sparse graph like Hamsterster, community detection methods Louvain and LGA have given better results than Kmeans. Kmedoids has outperformed for the C. Elegans dataset in every aspect, but Kmeans and LGA have also given impressive results. Again, we can observe from the perspective of the spectral norm (α) [36], i.e., the largest absolute eigenvalue of a network. The community detection algorithm works well on the higher value of α, e.g. Hamster and C. Elegans; except for the Jazz dataset, which has a high average degree and average number of edges. Kmeans has given the best results for the Jazz dataset.

Experiment 2: The average shortest path lengths (L_s) for separate proportion of initial influential spreaders (p), with λ = 1.5 is experimented in this case. We see in Fig: 3 for a lower value of (p), LGA or Louvain community detection algorithms have given better results, but if the percentage of first spreaders is high, the clustering algorithm works better in the maximum cases. So in the case of a larger dataset, if very few nodes are to be chosen as influential nodes, we should choose community algorithms over clustering methods, as these algorithms will choose nodes in a more scattered way. However, our algorithm has outperformed when compared with the benchmark methods in the domain.

VII. CONCLUSION

In this work, we show that the CancelOut method can be used for node ranking to identify influential spreaders along with GCN. Our method finds influential spreaders in the network, which gives better results when compared with other existing methods. Moreover, our experiments show the effects of different clustering and community detection algorithms on the spreading dynamics of influential nodes. Community detection methods work well when spreaders are from different communities, i.e., those spreaders will help to spread the news in that particular community. For further research, we focus on applying this technique on time-varying online social networks and find the influencers dynamically.

REFERENCES

[1] J.-X. Zhang, D.-B. Chen, Q. Dong, and Z.-D. Zhao, "Identifying a set of influential spreaders in complex networks," *Scientific reports*, vol. 6, no. 1, pp. 1–10, 2016.

[2] R. K. Karunakaran, S. Manuel, and E. N. Satheesh, *Spreading information in complex networks: an overview and some modified methods*. IntechOpen, 2017.

[3] M. Kitsak, L. K. Gallos, S. Havlin, F. Liljeros, L. Muchnik, H. E. Stanley, and H. A. Makse, "Identification of influential spreaders in complex networks," *Nature physics*, vol. 6, no. 11, pp. 888–893, 2010.

[4] A. Zeng and C.-J. Zhang, "Ranking spreaders by decomposing complex networks," *Physics Letters A*, vol. 377, no. 14, pp. 1031–1035, 2013.

[5] A. Namtirtha, A. Dutta, and B. Dutta, "Weighted kshell degree neighborhood: A new method for identifying the influential spreaders from a variety of complex network connectivity structures," *Expert Systems with Applications*, vol. 139, p. 112859, 2020.

[6] S. Manuel and K. R. Kumar, "An improved k-shell decomposition for complex networks based on potential edge weights," *Int. J of Applied Mathematical Sciences*, pp. 163–168, 2016.

[7] P. Liu, L. Li, S. Fang, and Y. Yao, "Identifying influential nodes in social networks: A voting approach," *Chaos, Solitons & Fractals*, vol. 152, p. 111309, 2021.

[8] F. Morone and H. A. Makse, "Influence maximization in complex networks through optimal percolation," *Nature*, vol. 524, no. 7563, pp. 65–68, 2015.

[9] D. Chen, L. Lü, M.-S. Shang, Y.-C. Zhang, and T. Zhou, "Identifying influential nodes in complex networks," *Physica a: Statistical mechanics and its applications*, vol. 391, no. 4, pp. 1777–1787, 2012.

[10] A. Namtirtha, A. Dutta, B. Dutta, A. Sundararajan, and Y. Simmhan, "Best influential spreaders identification using network global structural properties," *Scientific reports*, vol. 11, no. 1, pp. 1–15, 2021.

[11] A. Namtirtha, B. Dutta, and A. Dutta, "Semi-global triangular centrality measure for identifying the influential spreaders from undirected complex networks," *Expert Systems with Applications*, vol. 206, p. 117791, 2022.

[12] A. Saxena and S. Iyengar, "Centrality measures in complex networks: A survey," *arXiv preprint arXiv:2011.07190*, 2020.

[13] W. Chen, Y. Wang, and S. Yang, "Efficient influence maximization in social networks," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009, pp. 199–208.

[14] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003, pp. 137–146.

[15] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," *Advances in neural information processing systems*, vol. 29, 2016.

[16] J. Wang, V. W. Zheng, Z. Liu, and K. C.-C. Chang, "Topological recurrent neural network for diffusion prediction," in *2017 IEEE international conference on data mining (ICDM)*. IEEE, 2017, pp. 475–484.

[17] Z. Cao, K. Han, and J. Zhu, "Information diffusion prediction via dynamic graph neural networks," in *2021 IEEE 24th international conference on computer supported cooperative work in design (CSCWD)*. IEEE, 2021, pp. 1099–1104.

[18] S. Sharma and R. Sharma, "Identifying possible rumor spreaders on twitter: A weak supervised learning approach," in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021, pp. 1–8.

[19] W. Ye, Z. Liu, and L. Pan, "Who are the celebrities? identifying vital users on sina weibo microblogging network," *Knowledge-Based Systems*, vol. 231, p. 107438, 2021.

[20] M. Zhang, X. Wang, L. Jin, M. Song, and Z. Li, "A new approach for evaluating node importance in complex networks via deep learning methods," *Neurocomputing*, vol. 497, pp. 13–27, 2022.

[21] B. Fatemi, S. Molaei, S. Pan, and S. A. Rahimi, "Gcnfusion: An efficient graph convolutional network based model for information diffusion," *Expert Systems with Applications*, vol. 202, p. 117053, 2022.

[22] V. Borisov, J. Haug, and G. Kasneci, "Cancelout: A layer for feature selection in deep neural networks," in *International conference on artificial neural networks*. Springer, 2019, pp. 72–83.

[23] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[24] A. Paul and A. Dutta, "Community detection using local group assimilation," *Expert Systems with Applications*, vol. 206, p. 117794, 2022.

[25] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.

[26] 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, 2008.

[27] Y. Moreno, R. Pastor-Satorras, and A. Vespignani, "Epidemic outbreaks in complex heterogeneous networks," *The European Physical Journal B-Condensed Matter and Complex Systems*, vol. 26, no. 4, pp. 521–529, 2002.

[28] C. Castellano and R. Pastor-Satorras, "Thresholds for epidemic spreading in networks," *Physical review letters*, vol. 105, no. 21, p. 218701, 2010.

[29] P. M. Gleiser and L. Danon, "Community structure in jazz," *Advances in complex systems*, vol. 6, no. 04, pp. 565–573, 2003.

[30] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A.-L. Barabási, "The large-scale organization of metabolic networks," *Nature*, vol. 407, no. 6804, pp. 651–654, 2000.

[31] Q. Ni, J. Kang, M. Tang, Y. Liu, and Y. Zou, "Learning epidemic threshold in complex networks by convolutional neural network," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 29, no. 11, p. 113106, 2019.

[32] A. Sala, L. Cao, C. Wilson, R. Zablit, H. Zheng, and B. Y. Zhao, "Measurement-calibrated graph models for social network experiments," in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 861–870.

[33] D. Blakely, J. Lanchantin, and Y. Qi, "Time and space complexity of graph convolutional networks," *Accessed on: Dec*, vol. 31, 2021.

[34] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[35] C. Guo, L. Yang, X. Chen, D. Chen, H. Gao, and J. Ma, "Influential nodes identification in complex networks via information entropy," *Entropy*, vol. 22, no. 2, p. 242, 2020.

[36] J. Kunegis, "Handbook of network analysis," *arXiv preprint arXiv:1402.5500*, 2014.