DynamicG2B: Dynamic Node Classification with Layered Graph Neural Networks and BiLSTM

Fattah Muhammad Tahabi, Xiao Luo
Purdue School of Engineering and Technology
Indiana University-Purdue University Indianapolis
799 West Michigan Street
Indianapolis, IN 46202
ftahabi@iu.edu, luo25@iupui.edu

Abstract

Most studies in graph theory assume that graphs are static, but in reality, graph structures and features change over time, leading to the concept of dynamic graphs, which is an under-researched area. Contemporary research in dynamic graph representation learning typically treats different snapshots of the graph as separate entities, disregarding the benefits of incorporating temporal information. While some techniques try to solve this problem using recurrent neural network-based solutions, these approaches still face the challenge of the vanishing or exploding gradient problem and complicated training procedures. To address these issues, we propose DynamicG2B, a BiLSTM-based graph neural architecture that computes node representations guided by attention using neighborhood aggregation. Our method applies relevant attention weights at different time steps to classify nodes in a supervised manner, utilizing dynamic edges and node feature information. Our evaluation of two benchmark datasets shows that DynamicG2B outperforms seven state-of-the-art baseline models in node classification in dynamic graphs. Additionally, our analysis of attention weights opens up opportunities for further research into exploring the importance of relationships among graph nodes.

Introduction

The traditional approach to solving the dynamic node classification problems is to apply the static graph node classification algorithms to different time steps individually, then merge the results for the final prediction, or only apply the node classification to the last time step. These traditional approaches do not leverage the order of the information in graph topology and properties.

This work aims to tackle the challenge of dynamic node classification in graph-structured data, a difficult task due to the changing relationships and attributes of nodes over time. To address this issue, we introduce a new graph neural network-based solution, DynamicG2B, that combines graph convolutional networks and sequential processing to effectively capture the dynamic nature of graph data and improve the accuracy of node classification. DynamicG2B features two graph neural network layers: GraphSAGE and the Graph Attention Network (GAT). A BiLSTM layer is added on top of the GAT layer to sequentially process the nodes at each time step.

We carried out experiments on two public datasets to assess the performance of DynamicG2B and compare it with seven state-of-the-art models using evaluation metrics such as Accuracy, AUC, and F1 scores. The results show that DynamicG2B outperforms the other models. Additionally, an ablation study was performed to provide insight into the significance of each component of DynamicG2B. The code of DynamicG2B is released on Github[1].

The main contributions of this research include the following:

- We introduced a novel graph computation model for dynamic node classification;
- We compared the performance of our model against seven different state-of-the-art models to show the advantages of our model
- We demonstrated that using our model, we can interpret the dynamic edge importance using the attention weights of GAT.

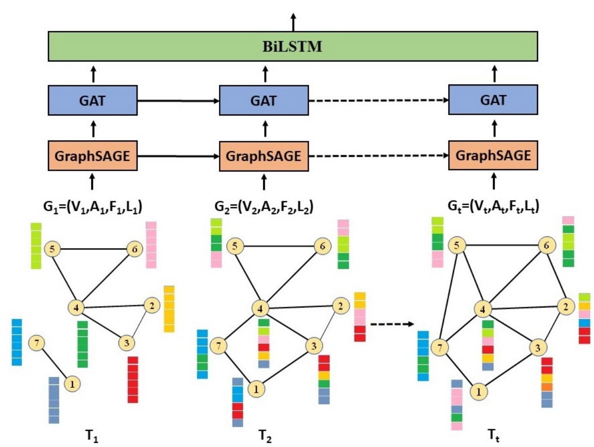

Figure 1: System Framework of DynamicG2B

Methodology

A dynamic graph consisting of snapshots $D = (G_1, G_2, G_3, ..., G_T)$, where $G_t = (V_t, A_t, F_t, L_t)$ rep-

[1]https://github.com/tahabi09/DynamicG2B-Dynamic-Node-Classification

resents the snapshot at timestamp t. The goal is to classify the class label L_T^{test} at time t in the test set given the training set class label L_T^{train}. The proposed model, DynamicG2B, is introduced for dynamic graph node classification, which leverages the graph attention mechanism to determine the significant relations among the vertices.

DynamicG2B Architecture Overview

The proposed DynamicG2B model addresses the issue of node classification in dynamic graphs with time-varying structures and properties. It consists of three interconnected layers: GraphSAGE [Hamilton, Ying, and Leskovec2017] for extracting graph structure information and node characteristics, Graph Attention Mechanism [Velickovic et al.2017] for simulating node relationships and taking attention weights of nodes into consideration, and Bidirectional Long Short-Term Memory for updating node embeddings and performing node classification. DynamicG2B uses a two-layer GraphSAGE for each timestamp, where the first layer captures first-hop neighbor information and the second layer considers two-hop neighbors. An attention mechanism is applied using a Graph Attention Network layer on the current output after the final layer convolution operation. The resulting node embeddings are then passed through a BiLSTM [Graves and Schmidhuber2005] layer to capture long-term dependencies between time steps. The utilization of the BiLSTM layer enhances the overall performance of the model, contributing to its success in accurately capturing and analyzing dynamic graph data.

Dynamic Edge Importance Interpretation

The process of determining the contribution of different edges at different timestamps involves utilizing the GAT layer. This technique replaces the normalizing convolution operation with the attention mechanism.

The result of this process is the computation of the vertex embedding $h_i^{(l+1)}$ in layer $(l+1)$ from the embeddings in layer l. The merging of the embeddings from the neighbors is achieved by multiplying the attention scores, as expressed in Equation 1.

$$h_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}(i)} \alpha_{ij}^{(l)} z_j^{(l)}\right) \tag{1}$$

Here, α_{ij} is the attention coefficient between nodes i and j. $N(i)$ represents all the neighbors of node i and σ is the activation function. z_i is the product of the learnable weight matrix W and lower layer embedding h as Equation 2.

$$z_i^{(l)} = W^{(l)} h_i^{(l)} \tag{2}$$

Pairwise unnormalized attention score e_{ij} between two neighbors i and j are shown in Equation 3, which first concatenates the z-embeddings of the two nodes.

$$e_{ij}^{(l)} = \text{LeakyReLU}(\vec{a}^{(l)^T}(z_i^{(l)}||z_j^{(l)})) \tag{3}$$

The symbol $||$ represents the concatenation operation in this context,. The result of this operation is then multiplied by a learnable weight vector $\vec{a}$ using the dot product. The result of the dot product is fed into a LeakyReLU activation function. The attention scores of the incoming edges of each vertex are normalized using a softmax function, as represented by Equation 4. This normalization provides the attention coefficients for each edge.

$$\alpha_{ij}^{(l)} = \frac{\exp(e_{ij}^{(l)})}{\sum_{k\in\mathcal{N}(i)} \exp(e_{ik}^{(l)})} \tag{4}$$

The attention mechanism in the DynamicG2B algorithm provides a way to determine the importance of the edges in a graph over time. The attention weight α is assigned to each edge to represent the significance of that edge at a particular timestamp. These attention weights are determined by taking into account the relationship between the nodes and other contextual information, such as the type of relationship or the length of time the relationship has existed.

Experimental Results

Datasets

The experiments carried out in this research aimed to comprehensively evaluate the effectiveness and efficiency of the proposed DynamicG2B model for dynamic node classification in graph-structured data. The performance of the DynamicG2B model was evaluated using two benchmark datasets, DBLP3 and DBLP5, as shown in Table-1, sourced from the computer science bibliography website DBLP[2] consisting of a extensive collection of catalog information from conferences and journals in different domains. The authors in the large collection of conferences and journals were represented as nodes in a graph, with co-authorship relationships determining the connections between nodes. The node attributes in the datasets were obtained using the word2vec algorithm [Mikolov et al.2013]. This technique extracted the properties of each node from the paper titles and abstracts for each year. The authors were divided into three and five classes, respectively, based on their research areas, which remained unchanged over time.

Table 1: Summary of Two Benchmark Real Datasets

Dataset	Nodes	Edges	Time Steps	Attributes	Classes
DBLP3	4257	23540	10	100	3
DBLP5	6606	42815	10	100	5

Baselines and Evaluation Metrics

We compared our DynamicG2B model to the following state-of-the-art baselines including GCN [Kipf and Welling2016], GAT [Velickovic et al.2017], GraphSAGE [Hamilton, Ying, and Leskovec2017] :

- GC-LSTM [Chen, Wang, and Xu2018]: GC-LSTM (Graph Convolution-LSTM) leverages both GCN and long-short term memory (LSTM) to model the dynamic

[2]https://dblp.org/

Table 2: Model Comparison with ACC, AUC, and F1

	DBLP3(70%)			DBLP3(50%)			DBLP5(70%)			DBLP5(50%)		
	ACC	AUC	F1	ACC	AUC	F1	ACC	AUC	F1	ACC	AUC	F1
GCN [Kipf and Welling2016]	74.3	**55.2**	63.6	70.1	54.4	65.0	57.1	49.9	47.1	54.3	51.7	45.1
GAT [Velickovic et al.2017]	67.3	53.8	54.3	73.5	54.7	64.3	62.8	50.7	51.4	63.8	52.8	52.4
GraphSAGE [Hamilton, Ying, and Leskovec2017]	71.9	54.1	60.5	72.9	52.2	64.1	64.6	**56.8**	57.0	61.7	53.3	52.8
GC-LSTM [Chen, Wang, and Xu2018]	74.4	51.3	64.2	70.5	51.7	62.2	60.5	50.8	47.6	65.6	47.7	54.0
EGCN [Pareja et al.2020]	51.2	53.2	49.9	50.8	50.9	47.5	39.8	47.8	35.0	38.4	50.6	32.9
RNNGCN [Yao and Joe-Wong2021]	70.1	46.2	58.1	71.1	52.4	65.4	57.9	49.3	51.0	62.5	49.2	53.5
TRNNGCN [Yao and Joe-Wong2021]	71.6	53.5	59.9	59.5	51.6	52.0	56.1	47.5	48.4	53.6	48.5	46.6
DynamicG2B (Our Model)	**81.0**	50.6	**73.1**	**80.3**	**54.9**	**68.6**	**71.5**	53.2	**59.7**	**68.7**	**55.0**	**55.2**

network structure and temporal dependencies, respectively. The combination of GCN and LSTM allows GC-LSTM to capture both the structural and temporal aspects of the dynamic network, making it a powerful tool for dynamic node classification tasks.

- EGCN [Pareja et al.2020]: EGCN extends GCN to handle dynamic graph inputs. It can continuously learn from new graph structures, resulting in improved performance and stability compared to retraining the GCN model every time the graph structure changes.
- RNNGCN and TRNNGCN [Yao and Joe-Wong2021]: The RNNGCN model uses a single decay rate as the RNN parameter. The model consists of two parts: an RNN and a two-layer GCN. The RNN component of the model learns the decay rate, and the GCN component uses this learned decay rate to cluster the weighted graphs. TRNNGCN (Transitional RNNGCN) is similar to RNNGCN, but instead of using a single decay rate, TRNNGCN uses a matrix of decay rates for different pairs of classes. The network replaces the decay method in RNNGCN with a new formula that incorporates the cluster predictions and the matrix of decay rates.

To evaluate the performances, we chose the standard accuracy (ACC), area under the ROC curve (AUC), and F1 which are also used in the baselines. The code of the baseline methods and datasets is publicly available [3].

Hyper-Parameter Setting

We created DBLP3(50%), DBLP5(50%), DBLP3(70%), and DBLP5(70%) datasets by splitting the datasets into training, validation, and test sets as 70%, 20%, 10%, and 50%, 20%, 30%, respectively. The baseline parameters are implemented using parameters listed in the paper [Yao and Joe-Wong2021]. For our DynamicG2B model, the Adam optimizer with a learning rate of 0.025 is used. The dropout rate was kept at 0.5. DynamicG2B uses two dropout layers in total, one between each of the hidden layers. Training epochs are set at 30.

Performance Results

The result shows that DynamicG2B shows better Accuracy, AUC, and F1 score than most of the baseline models, shown in Table-2.

[3]https://github.com/InterpretableClustering/InterpretableClustering

We agree that the AUC values around 0.5 may suggest that the classifiers are not significantly better than random guessing. However, it is important to note that the AUC values need to be interpreted in the context of the baseline performance and the available data. We reported the average AUC values over multiple runs and also conducted statistical tests to assess the significance of the differences between the methods.

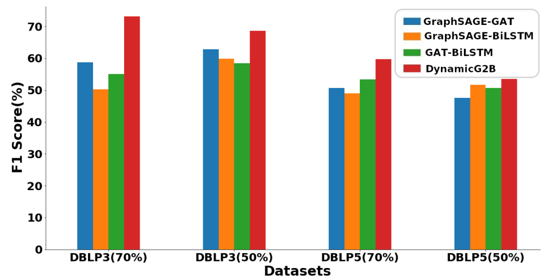

Figure 2: F1 of the Model Analysis

Ablation Study

We evaluated the impact of each component of DynamicG2B by conducting experiments with various components disabled. F1 score results are shown in Figures 2.

GraphSAGE-GAT: To evaluate the impact of the BiLSTM component, we disabled it and ran the model using only the GraphSAGE and GAT layers. This action removed the ability to incorporate temporal information from different snapshots. Processing time was faster without the BiLSTM component, but the model lost the ability to consider time-dependency.

GraphSAGE-BiLSTM: We measured DynamicG2B without the GAT layer, using only the GraphSAGE and BiLSTM layers. Results showed it could not account for varying node relationship weights or changing edge importance over time.

GAT-BiLSTM: We also test the performance of a model consisting of only the GAT and BiLSTM layers. In this model, the outputs from the GAT layers at different timestamps are fed into the BiLSTM layer. However, without the

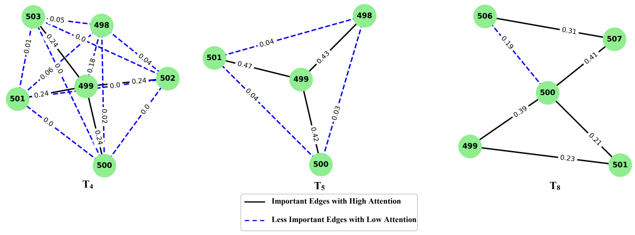

Figure 3: Attention Importance of Some Edges by DynamicG2B, Dataset: DBLP5(50%) (Significant Edges are Solid Black)

GraphSAGE layer, it lacks the ability to gather information from the neighbors of a node.

The comparison results depicted in figure 2 provide a comprehensive evaluation of the performance of different models on dynamic node classification tasks. The results demonstrate that the DynamicG2B model is the best-performing model in terms of F1 score, outperforming the other models.

Dynamic Edge Interpretation

The ability of the DynamicG2B model to anticipate edge importance at different timestamps is crucial. It leverages the power of attention coefficients obtained from the GAT layer.

To demonstrate the edge importance interpretation, we take the largest connected subgraphs with some highly ranked edges at timestamps T_4, T_5, and T_8 from the DBLP5(50%) dataset. In Figure 3, the threshold for each timestamp was set to 0.2, and edges with attention coefficients above this threshold were represented by solid black lines. In contrast, edges with comparatively lower attention coefficients, considered to be of less importance, were represented by blue dashed lines.

Through this process, edges $(499, 500)$, $(499, 501)$, and $(500, 501)$ were found to be present in the dynamic subgraph at each timestamp. The attention coefficients for edges $(499, 500)$ and $(499, 501)$ were consistently above the threshold and were determined to be the most significant and consistent across all timestamps. However, edge $(500, 501)$ showed high importance at timestamp T_8 despite having low attention coefficients at timestamps T_4 and T_5. This may be due to new connections from node 500 to nodes 506 and 507 being made at T_8. The DynamicG2B model effectively captured the importance of this edge at T_8 by considering the structure and attributes of these updated nodes and edges. The attention mechanism in DynamicG2B was able to differentiate between significant and insignificant edges, showcasing its ability to account for the context and structure of nodes and edges in the dynamic graph.

Related Work

Dynamic Graph Neural Networks (GNNs) have become increasingly popular in recent years for modeling evolving graph structures over time. This is due to their ability to capture the complex dependencies and interactions between nodes in a graph, as well as their ability to handle the dynamic nature of real-world data. The research in this field can mainly be categorized into two types: snapshot-based and event-based.

Snapshot-based methods process a fixed set of graph snapshots at discrete time intervals. These methods can be further divided into two categories:'Model Evolution' and 'Embedding Evolution'. Model Evolution methods, such as EvolveGCN [Pareja et al.2020], evolve the model parameters over time to capture the changes in the graph structure. On the other hand, Embedding Evolution methods, such as VGRNN [Hajiramezanali et al.2019], DySAT [Sankar et al.2020], and Roland [You, Du, and Leskovec2022], evolve the node embeddings over time to reflect the changes in the graph. In the case of snapshot-based approaches, after collecting the snapshots, they are aggregated by treating the representation as a static graph, which is then fed into a classification model [Skarding, Gabrys, and Musial2021] [Hisano2018] [Fan, Yao, and Joe-Wong2021].

Event-based methods process the graph as a sequence of edge addition and deletion events. These techniques can be split into two categories: 'Dynamic Embedding' and 'Dynamic Neighborhood'. Dynamic embedding methods, such as TGAT [Xu et al.2020], NAT [Luo and Li2022], and TGL [Zhou et al.2022], dynamically update node embeddings in response to changes in the graph. Dynamic Neighborhood methods, such as TGL [Zhou et al.2022], APAN [Wang et al.2021], DGNN [Ma et al.2020], and TGN [Rossi et al.2020], dynamically update the neighborhood of each node to reflect changes in the graph structure.

Conclusions and Future Work

In summary, the proposed DynamicG2B model offers a robust and interpretable solution for node classification in dynamic graphs. This model consists of two main components: A GAT layer that computes attention coefficients that emphasize the importance of edges, and a BiLSTM layer that encodes temporal information associated with nodes. The GAT layer and the BiLSTM layer allow the model to understand the evolving nature of the nodes and edges in the graph, resulting in better performance compared to the baseline.

Our future work includes (1) retrieving the significant sub-graphs from the large graphs, which might have a more significant impact on the whole graph throughout the timestamps, (2) computing and reporting ACC, AUC and F1 scores for each class in multiclass classification problems, (3) using more systematic experiments of variable percentage for a wide range of datasets for performance evaluation and ablation study, (4) applying such dynamic node classification towards solving real-world applications, such as disease prediction based on patient networks; and (5) extending DynamicG2B for multi-label classification on dynamic graphs.

References

Chen, J.; Wang, X.; and Xu, X. 2018. Gc-lstm: Graph convolution embedded lstm for dynamic link prediction. *arXiv preprint arXiv:1812.04206*.

Fan, Y.; Yao, Y.; and Joe-Wong, C. 2021. Gcn-se: Attention as explainability for node classification in dynamic graphs. In *2021 IEEE International Conference on Data Mining (ICDM)*, 1060–1065. IEEE.

Graves, A., and Schmidhuber, J. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks* 18(5-6):602–610.

Hajiramezanali, E.; Hasanzadeh, A.; Narayanan, K.; Duffield, N.; Zhou, M.; and Qian, X. 2019. Variational graph recurrent neural networks. *Advances in neural information processing systems* 32.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30.

Hisano, R. 2018. Semi-supervised graph embedding approach to dynamic link prediction. In *International Workshop on Complex Networks*, 109–121. Springer.

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

Luo, Y., and Li, P. 2022. Neighborhood-aware scalable temporal network representation learning. *arXiv preprint arXiv:2209.01084*.

Ma, Y.; Guo, Z.; Ren, Z.; Tang, J.; and Yin, D. 2020. Streaming graph neural networks. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, 719–728. New York, NY, USA: Association for Computing Machinery.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Pareja, A.; Domeniconi, G.; Chen, J.; Ma, T.; Suzumura, T.; Kanezashi, H.; Kaler, T.; Schardl, T.; and Leiserson, C. 2020. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 5363–5370.

Rossi, E.; Chamberlain, B.; Frasca, F.; Eynard, D.; Monti, F.; and Bronstein, M. 2020. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*.

Sankar, A.; Wu, Y.; Gou, L.; Zhang, W.; and Yang, H. 2020. Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In *Proceedings of the 13th international conference on web search and data mining*, 519–527.

Skarding, J.; Gabrys, B.; and Musial, K. 2021. Foundations and modeling of dynamic networks using dynamic graph neural networks: A survey. *IEEE Access* 9:79143–79168.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *stat* 1050:20.

Wang, X.; Lyu, D.; Li, M.; Xia, Y.; Yang, Q.; Wang, X.; Wang, X.; Cui, P.; Yang, Y.; Sun, B.; and Guo, Z. 2021. Apan: Asynchronous propagation attention network for real-time temporal graph embedding. In *Proceedings of the 2021 International Conference on Management of Data*, SIGMOD '21, 2628–2638. New York, NY, USA: Association for Computing Machinery.

Xu, D.; Ruan, C.; Korpeoglu, E.; Kumar, S.; and Achan, K. 2020. Inductive representation learning on temporal graphs. *arXiv preprint arXiv:2002.07962*.

Yao, Y., and Joe-Wong, C. 2021. Interpretable clustering on dynamic graphs with recurrent graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

You, J.; Du, T.; and Leskovec, J. 2022. Roland: Graph learning framework for dynamic graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, 2358–2366. New York, NY, USA: Association for Computing Machinery.

Zhou, H.; Zheng, D.; Nisa, I.; Ioannidis, V.; Song, X.; and Karypis, G. 2022. Tgl: A general framework for temporal gnn training on billion-scale graphs. *arXiv preprint arXiv:2203.14883*.