Temporal Adaptive Aggregation Network for Dynamic Graph Learning

Man Wu
Dept. of Electrical Engineering and Computer Science
Florida Atlantic University
Boca Raton, FL 33431, USA
mwu2019@fau.edu

Xingquan Zhu
Dept. of Electrical Engineering and Computer Science
Florida Atlantic University
Boca Raton, FL 33431, USA
xzhu3@fau.edu

2022 IEEE International Conference on Big Data (Big Data) | 978-1-6654-8045-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/BigData55660.2022.10020659

***Abstract*—Dynamic graphs are common in many applications, such as social networks with evolving nodes and edges over time. When handling such dynamics, existing approaches typically suffer from two limitations: (1) they primarily focus on network topology, without taking node class connections and temporal changes into consideration; and (2) the learning objective is primarily constrained by labeled nodes, which often result in over-smoothing and weak-generalization in representation learning, because labeled nodes are limited. In this paper, we propose a temporal adaptive aggregation network (`TAAN`) for dynamic graph learning. We consider a dynamic graph as a network with changing nodes and edges in temporal order. The temporal adaptive aggregation is to ensure that, for each node, the information aggregation is to consider neighbors from different classes, as well as their temporal order. For each snapshot of the dynamic network, data augmentation and consistency loss are combined to leverage labeled and unlabeled nodes to learn good node embedding. Meanwhile, in order to accommodate temporal changes of graphs, an incremental learning process is used to ensure that learning on each snapshot can inherit weights learned from previous time points, so graph learning can adapt to the dynamic graph environments. Experiments on real-world datasets validate the effectiveness of our approach.**

***Index Terms*—Temporal adaptive aggregation, temporal network, dynamic graph**

I. Introduction

Graphs are becoming fundamental tools for modeling complicated relationships in many applications. Instead of assuming samples are IID (Independent and Identically Distributed), graphs provide additional information through dependency relationships between objects, allowing a machine learning method to evaluate samples not only using their feature values, but also their topological relationships.

With the rapid development of networking platforms, the field o f g raph r epresentation l earning [1]–[4] h as gradually attracted widespread attention. The rapid development of graph neural networks (GNN) [5], [6] has led to great success in graph representation learning for tasks such as node classification [5], [7], l ink p rediction [8], a nd graph classification [9]. To date, most graph representation learning algorithms are applicable to static graphs, where learning and obtaining low-dimensional node embeddings can reflect the underlying node local structure. Nevertheless, many real-world applications involve dynamic or temporal graphs [10]–[14], which evolve over time. Simply applying algorithms designed for static graphs to dynamic graphs would make these algorithms underperform, because they cannot capture temporal information. Therefore, it is imperative to explore approaches applicable to dynamic graphs.

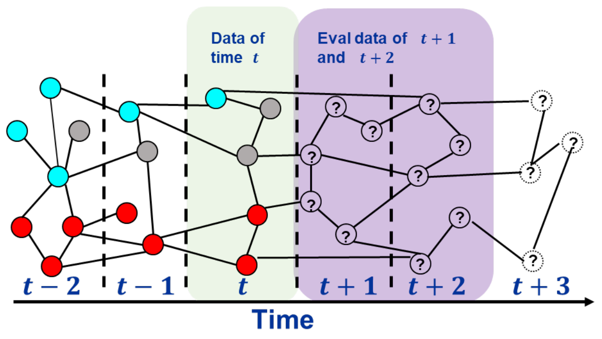


Fig. 1. A conceptual view of dynamic graph node classification where graph dynamically evolves over time. At each time t, the learner needs to classify vertices arriving in future time points, such as, $t+1$ and $t+2$, with maximum accuracy (Red and blue colored nodes are labeled). For best performance, a learner needs to leverage knowledge of subgraphs from previous moments to adapt to the current subgraph.

In this paper, we address a dynamic graph learning scenario that nodes and their relationships continuously evolve over time as shown in Figure 1. At each time point t, the learner needs to classify new vertices arriving in a near future, such as, time $t+1$ and time $t+2$, with maximum accuracy. Due to the dynamic evolving nature, the learner can use knowledge of subgraphs from past to adapt to the current subgraph for dynamic graph learning.

A handful of methods exist to handle dynamic graphs, by preprocessing dynamic graphs as a sequence of discrete (snapshots) graphs and using recurrent networks or attention mechanisms to combine intermediate representations extracted through GNN-based encoders [15]. For these methods, the number of nodes does not change in these discrete graphs. In addition, these methods mainly utilize graph diffusion as an alternative to neighborhood aggregation for dynamic graph learning. Therefore, they only utilize information of limited neighbors and do not consider node class connections and temporal changes.

Another drawback of existing dynamic graph neural nets is that they only consider labeled nodes, and usually suffer from the limitations of over-smoothing and weak-generalization. In reality, labeled nodes may be very limited in the network, making most methods easily overfit to the scarce label information. Most efforts to addressing this issue are focused on how to fully leverage the large amount of unlabeled data in the network. Recently, methods, such as MixMatch [16], GRAND [17], have been proposed to use data augmentation for regularized training, which have achieved great success in image classification tasks. This motivates us to apply the idea to facilitate dynamic graph learning.

In this paper, we study dynamic graph learning, where network nodes and edges are continuously changing in temporal order. The underlying key challenges are mainly threefold:

- *Challenge 1*: How can we capture graph structure information from changing neighbors for dynamic graph learning?
- *Challenge 2*: How to design an end to end framework to effectively leverage more unlabeled data to facilitate dynamic graph learning?
- *Challenge 3*: How to ensure that learning can inherit previous knowledge and adapt to dynamic changes of the graph?

To overcome the above challenges, we propose a novel temporal adaptive aggregation network (TAAN) for learning from dynamic graphs with changing nodes and edges in temporal order. For *Challenge 1*, we propose a novel temporal adaptive aggregation to strengthen attention to intra-class nodes and attenuates attention to inter-class nodes by considering neighbors from different classes, as well as their temporal order, which can capture the importance of each neighbor node to a target node for better understanding of nodes. For *Challenge 2*, we utilize a graph data augmentation and consistency loss to properly incorporate unlabeled data into the learning process, which can optimize the prediction consistency of unlabeled nodes across different data augmentations. In order to accommodate temporal changes of graphs (*Challenge 3*), an incremental learning process is employed to ensure that the learning on each snapshot can inherit weights learned from previous time points, so graph learning is essentially adaptive to the dynamic graph environments. Experimental results on real datasets validate the design and effectiveness of our approach.

Comparing to existing work in the field, our contributions can be summarized as follows:

- We propose a temporal adaptive aggregation network (TAAN) for dynamic graph learning by considering a dynamic graph as a network with changing nodes and edges over time.
- We propose a novel temporal adaptive aggregation network for graph data, which can effectively utilize neighbors from different classes, as well as their temporal order.
- We propose a graph data augmentation and consistency loss to properly incorporate unlabeled data into the learning process, which can optimize the prediction consistency of unlabeled nodes across different data augmentations to learn good node embedding.
- Experiments on two dynamic graph datasets demonstrate that our graph neural network approach outperforms the baseline methods.

II. Problem Statement

Graph: A graph is represented as $G = (V, E, X, Y)$, where $V = \{v_i\}_{i=1,\cdots,N}$ is a vertex set representing the nodes in a graph, and $e_{i,j} = (v_i, v_j) \in E$ is an edge indicating the relationship between two nodes. The topological structure of graph G is represented by an adjacency matrix A, where $A_{i,j} = 1$ if $(v_i, v_j) \in E$; otherwise $A_{i,j} = 0$. $x_i \in X$ indicates content features associated with each node v_i, $y_i \in \mathcal{Y}$ denotes class label of v_i. $Y \in \mathbb{R}^{N \times C}$ is a binary (one-hot) label matrix of G, where N is the number of nodes in G and C is the number of node classes/categories (*i.e.* $|\mathcal{Y}| = C$). If a node $v_i \in V$ is associated with label l , $Y^l_{(i)} = 1$; otherwise, $Y^l_{(i)} = 0$.

Dynamic Graph Learning: A dynamic graph $\mathbb{G} = G_1, ...G_T$ consists of a number of snapshot subgraphs at different time point t, $G_t = (V_t, E_t, X_t, Y_t)$. Assume $V = L \bigcup U$, where L are the labeled nodes ($\forall v_i \in L$, $y_i \in \mathcal{Y}$) and U are unlabeled nodes. m and n denote the number of the labeled nodes and all nodes, respectively. At any time t, given a graph $G_t = (V_t, E_t, X_t, Y_t)$, we aim to learn a classifier model, $f_t : (A_t, X_t; L_t) \mapsto Y_t$, to predict the class labels for the unlabeled nodes U_t.

III. Proposed Method

In order to tackle above dynamic changes, we propose a framework, as shown in Figure 2, which employs an incremental learning process at each time point t to learn and update graph neural network models, such that the overall system can adapt to dynamics in the network. In our research, we consider dynamics and changes as variance of the network, and our main theme is to generate multiple graph data augmentations, perform feature aggregation and consistency regularization to learn best node features for each time step t.

A. Temporal Adaptive Aggregation Network

In traditional graph convolutional neural networks, the aggregation of the information (for node representation learning) is primarily driven by the network topology, *e.g.* using adjacency matrix A combined with feature matrix X to learn node embedding. In dynamic graphs, the topology of the network is continuously evolving, so our temporal adaptive aggregation network (TAAN) is designed to allow each node to aggregate information, by differentiating nodes from different classes as well as nodes arrived in different temporal orders. This allows TAAN to focus on changes of the network, instead of just driven by topology for embedding learning.

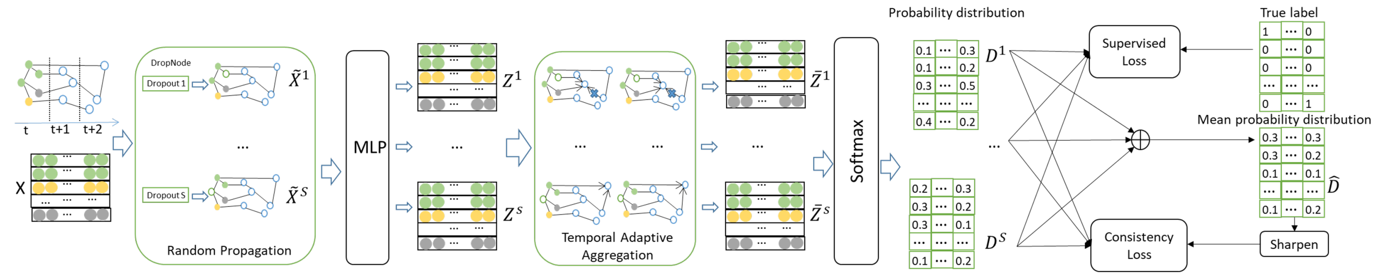


Fig. 2. The overall architecture of the proposed temporal adaptive aggregation network (TAAN). For subgraphs of task $\hat{\tau}_t$, TAAN uses a novel temporal adaptive aggregation network to strengthen attention to intra-class nodes and attenuates attention to inter-class nodes by considering the temporal and category information of nodes jointly, and consistency regularized learning is utilized to optimize the prediction consistency of unlabeled nodes across different data generations.

At time t, given an input graph G_t with its adjacency matrix A_t and feature matrix X_t, we utilize random propagation [17] to efficiently augment the graph data, wherein each node's features can be randomly dropped either partially (Dropout) or entirely, after which the perturbed feature matrix is propagated over the graph. For each augmentation $\tilde{X}^s$, it is then fed into a two-layer MLP, in order to obtain new node features Z^s. The MLP model can also be replaced with more complex and advanced GNN models, such as GCN and GAT.

In real-world graphs, some nodes may have neighbors that may be irrelevant to them or belong to different classes. Therefore, in the aggregation of neighbor nodes of the target node, these nodes of different classes will have an impact on the final representation of the target node, and the impact is negative, which will worsen the representation result. Therefore, when we do aggregation, we need to adjust the aggregation degree adaptively according to the local class context of each node. For each augmentation graph $\tilde{X}^s$ and the new node features Z^s, we assume that starting from node v_i, we determine the next node among the neighbor by comparing the class likelihood given the node features [18]. Here, we compare $\mathbf{p}_i = \bar{p}(\mathcal{Y}|z_i)$ and $\mathbf{p}_j = \bar{p}(\mathcal{Y}|z_j)$ for $j \in N(i)$. Our design objective is that the more similar $\mathbf{p}_i$ and $\mathbf{p}_j$, the more likely the walker moves from v_i to v_j.

For each augmentation graph, we can define the transition probability from v_i to v_j as Eq. (1).

$$M^s_{i,j} = \text{Softmax}_{j \in N(i)}(\mathbf{p}_i^T \mathbf{p}_j), \tag{1}$$

Let a row vector $\pi_i^{(k)} \in \mathbb{R}^{1\times N}$ be the state distribution after k steps. This can be naturally derived by a Markov chain, i.e., $\pi_i^{(k+1)} = \pi_i^{(k)} M^s$, where the initial state distribution $\pi_i^{(0)}$ be a one hot vector indicating the starting node v_i. So we can get the updated node representation as Eq.(2).

$$z_i^{(AA,s)} = \sum_j \pi_i^{(K)}(j) \cdot z_j^s, \tag{2}$$

where $z_i^{(AA,s)}$ is the new node representation of v_i that aggregates the neighbor node information for the augmentation graph $\tilde{X}^s$, and $\pi_i^{(K)}(j)$ is zero for v_j beyond K-hop from v_i. Hence, $\pi_i^{(K)}(j)$ can naturally reflect the class similarity as it grows with the similarity between $\mathbf{p}_i$ and $\mathbf{p}_j$.

During the aggregation process, for node v_i, we only aggregate the K-hop neighbor nodes that exist at the previous time $t-1$ and the current time t, and exclude the K-hop neighbor nodes that exist at the future time $t+1$. Therefore, $z_i^{(AA,s)}$ is essentially an attentive aggregation of K-hop neighbors which can strengthen attention to intra-class nodes and attenuates attention to inter-class nodes. Finally, we can form a new feature representation of the node v_i as follows:

$$z_i^{(TAA,s)} = (1-\gamma_i)\cdot z_i^s + \gamma_i \cdot z_i^{(AA,s)}, \tag{3}$$

where $z_i^{(TAA,s)}$ denotes the updated node feature by using the temporal adaptive aggregation module, and $\gamma_i \in [0,1]$ controls the trade-off between its own node feature z_i and the aggregated feature $z_i^{(AA,s)}$ in Eq. (2) by considering the local class-context of v_i. For the node with neighbors of the same class, γ_i should be a large value to better aggregate information of neighbors to obtain a better feature representation. For the node with neighbors of a different class, γ_i should be adjusted to a small value to preserve its original features and avoid the interference of irrelevant neighbor information. For simplicity, we define a control variable h_i as follows:

$$h_i = \frac{1}{deg(i)} \sum_{j\in N(i)} (\mathbf{p}_i^T \mathbf{p}_j), \tag{4}$$

where $deg(i)$ is the degree of v_i and the range of h_i would be $0 \le h_i \le 1$. Here, the greater the value of h_i means that the more nodes of the same category in the neighborhood, and vice versa. Therefore, we use an adaptive formula for γ_i as Eq. (5)

$$\gamma_i = (1-\beta)h_i + \beta\gamma_u, \tag{5}$$

where γ_u is set to 1 and it ensures the range of γ_i to be $0 \le \gamma_i \le 1$. $\beta \in [0,1]$ is a hyperparameter which balances the h_i and γ_u, and it can be determined empirically for each dataset since different graphs can capture different neighborhood information. Note that, for each augmentation graph $\tilde{X}^s$, we can obtain the new node representation $\bar{Z}^s$ by the proposed temporal adaptive aggregation module. Here, the network of each augmentation graph learns independently from each other and does not share. Therefore, we can obtain S new node features $\bar{Z}^s, s \in [1, ..., S]$.

B. Consistency Regularized Learning

In the dynamic graph learning, we aim to smooth the label information over the graph with regularizations. Therefore, we use a combination of the supervised loss on the labeled nodes and the graph consistency regularized loss. Given the $\bar{Z}^s$ generated in the Temporal Adaptive Aggregation Module for each augmentation graph, we normalize it using Softmax to get $D^s, s \in [1, ..., S]$.

1) Supervised Loss: The supervised objective of the graph node classification task is defined as the average cross-entropy loss over S augmentations:

$$\mathcal{L}_{sup} = -\frac{1}{S}\sum_{s=1}^{S}\sum_{i=0}^{m-1} Y_i^T \log D_i^s, \tag{6}$$

where m denotes the number of labeled nodes.

2) Consistency Regularization Loss: For the unlabeled data, we utilize a prediction consistency loss and optimize it among S augmentations. For example, in our experiment, we have considered a simple case of S =2, we can minimize the squared L_2 distance between the two outputs, i.e.$min\sum_{i=0}^{n-1}||{D_i}^{(1)} - {D_i}^{(2)}||$. For the case where S is greater than 2, we first calculate the center of the distribution of labels by taking the average of all distributions, i.e., $\hat{D}_i = \frac{1}{S}\sum_{s=1}^{S}{D_i}^{(s)}$. And then followed by the work of [16], [17], we utilize a trick called "sharpening" to predict the labels based on the average distributions. Here, the i^{th} node's predicted probability on the c^{th} class can be calculated by:

$$\hat{D}_{i,c}^{'} = \frac{\hat{D}_{i,c}^{\frac{1}{\delta}}}{\sum_{c=0}^{C-1}\hat{D}_{ic}^{\frac{1}{\delta}}}, \quad 0 \le c \le C-1 \tag{7}$$

where $0 < \delta \le 1$ acts as the "temperature" controlling the "sharpness" of the category distribution. As $\delta \to 0$, the sharpened label distribution will approximate to the one-hot distribution. We minimize the distance between $\hat{D}_i^{'}$ and D_i in the following equation:

$$\mathcal{L}_{con} = \frac{1}{S}\sum_{s=1}^{S}\sum_{i=0}^{n-1}||\hat{D}_i{}' - {D_i}^{(s)}||_2^2, \tag{8}$$

So, by setting δ as a small value, we can force the entropy of this model to be small. This can be seen as adding an additional entropy minimization regularization to the model, which assumes that the decision boundary of the classifier should not pass through the high density region of the marginal data distribution. The final loss function is defined in Eq. (9), where λ is a hyper-parameter balancing betwee the two losses.

$$\mathcal{L} = \mathcal{L}_{sup} + \lambda\mathcal{L}_{con}, \tag{9}$$

TABLE I
STATISTICS OF TWO DYNAMIC NETWORKS.

Dataset	Node	Edges	Classes	Features	T
DBLP-easy	45,407	112,131	12	2,278	7
DBLP-hard	198,675	643,734	31	4,043	6

C. Incremental Learning

We use a sliding window to take the subgraph at each moment. For example, at time t, our subgraph timestamps include t, $t+1$, and $t+2$. In the process of acquiring each subgraph, the sliding step size and the number of classes remain the same. We adopt an incremental learning approach to initialize the training parameters on time t with the final parameters of the previous time θ^{t-1}.

IV. EXPERIMENTS

In this section, we will first describe experiment settings including datasets, baselines and experimental setup, experimental results and then report the parameter analysis.

A. Experiment Settings

1) Benchmark Datasets: We employ two newly compiled citation graph datasets based on DBLP (DBLP-easy and DBLP-hard) for dynamic graph learning [10]. For DBLP, we use the conferences and journals of the published papers as classes, and vertex features are normalized TF-IDF representations of the publication title.

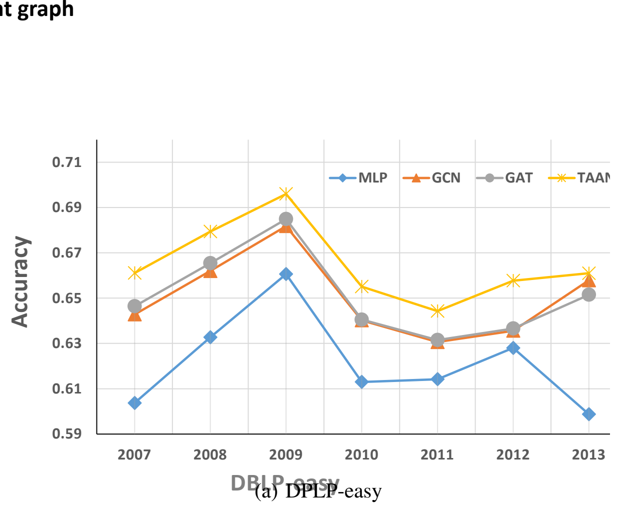
For each dataset, we construct the tasks of subgraph $\hat{\tau}_1, \cdots, \hat{\tau}_T$ on the basis of the publication year along with a time window size w. For each task $\hat{\tau}_t$, we construct a graph with respect to publications from time $[t, t+w]$, where publications from time t are the training vertices, and $[t+1, t+w]$ are test vertices. For example, in DBLP-easy, we use the data from 2007-2015, when the $w = 2$, there are a total of 7 subgraphs, namely 2007-2008-2009,$\cdots$,2013-2014-2015. For DBLP-hard, it has a larger number of nodes, edges, and categories. We use data from 1993-2000, when the $w = 2$, and there are a total of 6 subgraphs, namely 1993-1994-1995,$\cdots$,1998-1999-2000. Therefore, for each subgraph task $\hat{\tau}_t$, we can learn a classifier model,

$$f_t : (A_t, X_t; L_t) \mapsto Y_t \tag{10}$$

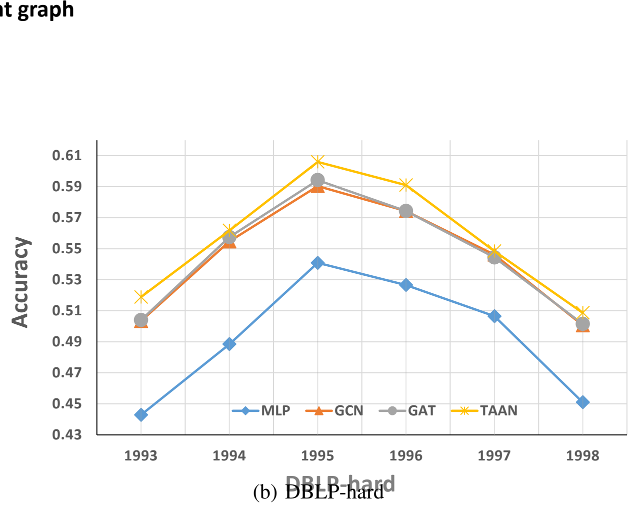
and use the learned classifier $f_t(\cdot)$ to predict class labels of the unlabeled nodes U_t. Finally, we can aggregate accuracy scores over the sequence of tasks $\hat{\tau}_1, \cdots, \hat{\tau}_T$. The details are shown in Table I.

2) Baseline Methods: We use following baselines with necessary adaption.

- **MLP** only uses a multi-layer perceptron (MLP) which only utilizes node features to train the model.
- **GCN** uses Graph Convolutional Network [5] for graph data, which integrates network topology, node features and observed labels into an end-to-end learning framework.



(a) DPLP-easy



(b) DBLP-hard

Fig. 3. Classification accuracy with respect to different time points.

TABLE II
THE AVERAGE ACCURACY ON DBLP.

Method	DBLP-easy	DBLP-hard
MLP	0.6216	0.4927
GCN	0.6501	0.5449
GAT	0.6510	0.5460
TAAN-c	0.6541	0.5499
TAAN-t	0.6611	0.5530
TAAN-i	0.6471	0.5519
TAAN	**0.6649**	**0.5558**

TABLE III
THE ACCURACY OF DIFFERENT SAMPLE NUMBERS S ON DBLP.

S	DBLP-easy	DBLP-hard
1	0.6603	0.5510
2	0.6639	0.5522
3	0.6643	0.5555
4	0.6649	0.5558
5	0.6649	0.5547

- **GAT** uses Graph Attention Network [19] for graph data.
- **TAAN-c** removes the Consistency Regularization loss in our proposed TAAN model, and only uses the temporal adaptive aggregation network and supervised loss.
- **TAAN-t** removes the Temporal Adaptive Aggregation Network in our proposed TAAN model, and only uses the supervised loss and consistency regularization loss.
- **TAAN-i** removes the incremental learning strategy in our proposed TAAN model, and re-initialize the training parameters on each task of subgraph and retrain from scratch.
- **TAAN** is our proposed model.

3) Experimental Setup: For fairness of comparison, for each subgraph task $\hat{\tau}_t$, nodes from time t are the training vertices, and $[t+1, t+w]$ are test vertices. Here, w is set to 2. We aggregate accuracy scores over the sequence of tasks $\hat{\tau}_1, ..., \hat{\tau}_T$ by using their unweighted average:

$$Acc(f) = \frac{1}{T} \sum_{t \in 1,...,T} Acc_t(f^{(t)}) \tag{11}$$

Here, we also conduct 5 trials of randomly splitting with different random seeds, and report the average **Accuracy** for each dataset as final experimental results.

All models were implemented in PyTorch with the Adam optimizer with a learning rate of $1e^{-3}$ for 2000 steps. We set the embedding dimension of nodes to 64 for all methods. We choose two layers for MLP, GCN and GAT, where the embedding dimension of the hidden layer is set to 64. The K in temporal adaptive aggregation network is set to 6, and the hyperparameters λ and β are set to 1 and 0.8, respectively. The δ for sharpness is set to 0.5. For TAAN, the number of graph data augmentations is 4 by default.

B. Experimental Results

We report the average accuracy of our model and baseline methods on two benchmark datasets in Table 2 and Figure 3. From results in Table 2 and Figure 3, we can conclude that:

(1) The GCN and GAT obtain better performance than MLP, both in terms of the average accuracy and specifically in terms of the accuracy at each time point. For the average accuracy, on the DBLP-easy dataset, GCN and GAT improved over MLP by 2.85% and 2.94%, respectively; On the DBLP-hard dataset, GCN and GAT are 5.22% and 5.33% higher than MLP, respectively. These results shows that graph neural networks can better learn node representations by leveraging node relationships.

(2) TAAN achieves better performance than GCN and GAT in all experimental results. In terms of average accuracy, on DBLP-easy dataset, TAAN is 1.48% and 1.39% higher than GCN and GAT, respectively; On DBLP-hard dataset, TAAN improves over GCN by 1.09% and GAT by 0.98%. In terms of the accuracy of each specific year, in 1995 on the DBLP-hard dataset, for example, TAAN improves the accuracy by about 1.3% compared with GCN and by about 1.02% compared with GAT. This phenomenon indicates the superior performance of the proposed temporal adaptive aggregation network, which

strengthens attention to intra-class nodes and attenuates attention to inter-class nodes by considering the temporal and category information of nodes jointly. Furthermore, consistency regularized learning is used to leverage labeled and unlabeled nodes to learn good node embedding.

(3) The TAAN obtains better performance than TAAN-t, which indicates that the Temporal Adaptive Aggregation network can not only aggregate the information of neighbor nodes, but also the temporal relationships of nodes.

(4) The performance of TAAN is better than that of TAAN-c, indicating that the introduction of the consistency regularization loss is effective.

(5) The performance of TAAN is better than that of TAAN-i, indicating that the effectiveness of the incremental learning. Here, in terms of average accuracy, on DBLP-easy and DBLP-hard dataset, TAAN can achieve relative improvements of 1.78% and 0.39% respectively, compared with TAAN-i.

C. Parameter Analysis

Analysis of results for each subgraph To further demonstrate the effectiveness of our method, we present the comparison results of different subgraphs at each time t for both datasets as shown in Figure 3(a) and Figure 3(b). From the results, we can see that our proposed TAAN is better than other baseline methods in different time periods, thus validating the effectiveness of our method again.

Analysis of the Number of graph data augmentation S The results for different numbers of graph data augmentation S are shown in Table III. It can be seen from the results that the larger the numbers of graph data augmentation S is, the better the experimental results will be. Here, we use the augmented graphs to explore consistency for more effective learning and prediction, and a consistency loss is used to incorporate unlabeled data into the learning process, and optimize the prediction consistency of unlabeled nodes across different data augmentations. However, as S becomes larger, the complexity of the experiment increases. In order to balance the efficiency and performance, we finally set S equal to 4.

V. Conclusions

Many applications, such as social networks and citation networks, involve networks with changing nodes and edges over time. In this paper, we study dynamic graph learning, where network nodes and edges are continuously changing in temporal order. We argued that when handling dynamics in the networks, existing approaches primarily focus on network topology, without taking node class connections and temporal changes into consideration, and their learning objective is primarily constrained by labeled nodes. To address the challenges, we proposed a temporal adaptive aggregation network (TAAN) for dynamic graph learning. The temporal adaptive aggregation jointly exploits neighbors from different classes, as well as their temporal order. Data augmentation and consistency loss are further integrated to leverage labeled and unlabeled nodes in the network. As a result, TAAN can adapt to changes in dynamic graphs to learn node embedding features for classification. Results on real-world datasets demonstrate the effectiveness of our algorithm.

Acknowledgment

This research is sponsored by US National Science Foundation under grant No. IIS-1763452.

References

[1] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Network representation learning: A survey," *IEEE Trans. on Big Data*, vol. 6, no. 1, pp. 3–28, 2020.

[2] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[3] M. Wu, S. Pan, and X. Zhu, "Openwgl: Open-world graph learning," in *20th IEEE International Conference on Data Mining, ICDM 2020, Sorrento, Italy, November 17-20, 2020*, pp. 681–690.

[4] M. Wu, S. Pan, C. Zhou, X. Chang, and X. Zhu, "Unsupervised domain adaptive graph convolutional networks," in *WWW '20: The Web Conf. 2020, Taipei, Taiwan, April 20-24, 2020*, 2020, pp. 1457–1467.

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

[6] S. Pan, R. Hu, S.-f. Fung, G. Long, J. Jiang, and C. Zhang, "Learning graph embedding with adversarial training methods," *IEEE Transactions on Cybernetics*, vol. 50, no. 6, pp. 2475–2487, 2020.

[7] M. Wu, S. Pan, L. Du, I. W. Tsang, X. Zhu, and B. Du, "Long-short distance aggregation networks for positive unlabeled graph learning," in *Proc. of ACM CIKM International Conference*, 2019.

[8] C. Wang, S. Pan, R. Hu, G. Long, J. Jiang, and C. Zhang, "Attributed graph clustering: A deep attentional embedding approach," in *Proc. of IJCAI*, 2019, pp. 3670–3676.

[9] H. Gao and S. Ji, "Graph u-nets," in *International Conference on Machine Learning*, 2019, pp. 2083–2092.

[10] L. Galke, B. Franke, T. Zielke, and A. Scherp, "Lifelong learning of graph neural networks for open-world node classification," in *International Joint Conference on Neural Networks, IJCNN 2021, Shenzhen, China, July 18-22, 2021*. IEEE, 2021, pp. 1–8.

[11] J. Leskovec, J. M. Kleinberg, and C. Faloutsos, "Graphs over time: densification laws, shrinking diameters and possible explanations," in *Proc. of the 11th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining, August 21-24, 2005*, 2005, pp. 177–187.

[12] M. Shi, Y. Huang, X. Zhu, Y. Tang, Y. Zhuang, and J. Liu, "Gaen: Graph attention evolving networks," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2021, pp. 1541–1547.

[13] R. Trivedi, M. Farajtabar, P. Biswal, and H. Zha, "Dyrep: Learning representations over dynamic graphs," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[14] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. M. Bronstein, "Temporal graph networks for deep learning on dynamic graphs," *CoRR*, vol. abs/2006.10637, 2020.

[15] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. B. Schardl, and C. E. Leiserson, "Evolvegcn: Evolving graph convolutional networks for dynamic graphs," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, 2020, pp. 5363–5370.

[16] D. Berthelot, N. Carlini, I. J. Goodfellow, N. Papernot, A. Oliver, and C. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2019, pp. 5050–5060.

[17] W. Feng, J. Zhang, Y. Dong, Y. Han, H. Luan, Q. Xu, Q. Yang, E. Kharlamov, and J. Tang, "Graph random neural networks for semi-supervised learning on graphs," in *Annual Conf. on Neural Information Processing Systems (NeurIPS)*, 2020.

[18] J. Lim, D. Um, H. J. Chang, D. U. Jo, and J. Y. Choi, "Class-attentive diffusion network for semi-supervised classification," in *Thirty-Fifth AAAI Conference on Artificial Intelligence, Virtual Event, February 2-9, 2021*, 2021, pp. 8601–8609.

[19] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.