MLCDG: Multi-Level Contrastive Graph Clustering in Dynamic Graphs

Mohamed Mahmoud Amar, Mohamed Bouguessa, and Abdoulaye Banire Diallo

University of Quebec at Montreal, Montreal, QC, Canada
`amar.mohamed_mahmoud@courrier.uqam.ca`
`bouguessa.mohamed@uqam.ca`
`diallo.abdoulaye@uqam.ca`

Abstract. Dynamic graphs are widespread in social networks, biological networks, and recommendation systems. Community detection in dynamic graphs presents several challenges because these graphs continually change over time. Traditional methods often tackle the problem by clustering graphs separately at each timestep and matching these communities. Unfortunately, this strategy does not account for the temporal continuity and can lead to instability in community detection. Moreover, most current methods do not consider the scenario where nodes have attributes. To address these challenges, we introduce a multi-level contrastive graph clustering approach for Dynamic Graphs (MLCDG), a novel methodology employing deep clustering in the context of dynamic graph neural networks. MLCDG innovatively incorporates contrastive learning to generate clustering-oriented latent representations that effectively capture both node-level and temporal-level community structures. Our approach stabilizes community detection by maintaining temporal coherence and minimizing clustering disruptions caused by dynamic changes. The methodology outperforms existing state-of-the-art techniques, demonstrated by our experiments on real-world and synthetic datasets, validating our approach's effectiveness in enhancing community detection in attributed dynamic graphs.

Keywords: Dynamic Graphs · Deep Clustering · Contrastive Learning · Community detection.

1 Introduction

Many real-world graphs, such as those in social networks [31], biological networks [10], and recommendation systems [1], commonly exhibit community structure. Community detection is essential for understanding real-world graphs, identifying important nodes, grouping similar nodes, and predicting future behavior. Despite the attention given to community detection in static graphs, relatively less research has focused on the dynamic scenario in which communities evolve. This problem is particularly challenging as slight changes can significantly impact the results of graph clustering methods [2]. Community detection in dynamic graphs is commonly approached as a problem with two distinct steps. First, the graphs are clustered independently at each time step using a static clustering algorithm. Then, the communities discovered at different time steps are matched [4], [6], [39]. This set of approaches is based on the assumption that communities at a time step are influenced solely by the state of the graph at that particular time. However, the inherent instability of clustering algorithms can hurt the quality of clustering results [2]. Discerning between changes in the community structure stemming from community evolution and those arising from algorithm instability can be challenging.

Several solutions have been suggested to address, or at least alleviate, the instability problem. They all share the goal of creating a more gradual evolution of communities. One commonly employed strategy for addressing instability involves iteratively leveraging historical information to identify the community structure at the time-step of interest. These methods allow us to identify communities at a specific point in time without diverging significantly from the community structure seen in previous time steps. In [2], the authors modified the Louvain [3] algorithm to stabilize the community structure. Another research direction [24, 8] explored two distinct facets. The first aspect is the current partition's quality, while the second is temporal coherence. This research initiative focuses on optimizing a quality function that integrates both aspects. Lin *et al.* [24] formulated the problem through non-negative matrix factorization, which unifies communities' factorization and temporal evolution. In [8], Folino *et al.* proposed a multi-objective evolutionary approach based on genetic algorithms. Unfortunately, all these meethods remain shallow at their core, heavily depending on similarity measures built around the topological similarities of the clusters, making them unreliable in the case of attributed graphs.

We argue that these methods fail to provide the expressive capabilities of neural networks. Neural networks, specifically graph neural networks, have demonstrated unprecedented achievements in learning graph representations by capturing information from the graph structure using message passing [37]. Over the past few years, deep clustering [40], [13] has exhibited remarkable effectiveness in grouping a dataset into clusters. This paradigm consists of two training phases: pretraining for solving a pretext task and fine-tuning based on pseudo labels. The objective of the fine-tuning phase is to transform the latent space so that it becomes clustering-oriented. This clustering approach has been employed with different types of datasets, including graph data [18], [28]. Despite deep learning achieving notable success in dynamic graphs across several downstream tasks, including node classification and edge prediction [26], [35], [23], [19], the application of deep clustering techniques for community detection in dynamic graphs remains unexplored.

Contrastive learning [25] has emerged as an effective approach for learning representations in machine learning, especially within unsupervised or self-supervised contexts. The goal of contrastive learning is to learn a meaningful representation of data by encouraging the model to pull similar data points closer together in the latent space while pushing dissimilar data points farther apart. This helps in creating a more discriminative and structured latent representation of the data. In unsupervised learning, negative and positive pairs are created by applying data augmentation techniques to generate multiple views of the same data point. One advantage of using contrastive learning with graphs is the capability to customize the learning approach to focus on a desired level of abstraction (node, cluster, time).

The problem of instability has been explored in [2]. The researchers selected three traditional static clustering algorithms and assessed their outcomes' stability when the network undergoes minor modifications. To achieve this, they compared the modularity and Normalized Mutual Information (NMI) scores across consecutive outcomes. In a network comprising 9,377 nodes and 24,107 edges derived from a co-authorship network, eliminating a single node resulted in NMIs between successive outcomes of 0.99, 0.9, and 0.75 for each algorithm, respectively. The impact becomes even more evident when considering the number of individual changes: a solitary node modification led to approximately 500, 2,000, and 3,000 individual community changes for the respective algorithms. Several solutions have been proposed to address, or at least alleviate, the instability issue. They all share the goal of creating a more gradual evolution of communities.

We introduce a novel method for clustering dynamic graphs, which we call Multi-Level Contrastive Graph Clustering for Dynamic Graphs (MLCDG). Our deep clustering approach uses contrastive learning principles to train a Dynamic Graph Neural Network (DGNN) to learn clustering-oriented latent representations. Specifically, we present three contrastive loss components that constitute the overall loss function to be minimized. During the pretraining phase, the loss function comprises two terms. The first term consists of node-level contrastive learning, which is used to learn a general-purpose graph representation. The second component is a temporal-level contrastive loss, designed to enforce a temporal smoothness constraint and address the community instability problem. During the clustering phase (second phase), a novel loss term is incorporated to refine the clusters identified at the end of the pretraining stage.

Contributions. (1) We establish the first deep clustering model for dynamic graphs that tackles the instability problem in static community detection algorithms. **(2)** We introduce a novel multi-level contrastive loss function to improve node representation learning by harnessing data from multiple abstraction levels. **(3)** In the fine-tuning stage, we incorporate a contrastive clustering loss aimed at gradually transforming the latent space to become clustering-oriented. This transformation capitalizes on the nodes assigned with high confidence. We conduct experiments on real-world and synthetic datasets. The results illustrate that the proposed model exceeds state-of-the-art methods' performance.

2 Related Work

Graph Neural Networks (GNNs) have gained significant attention and demonstrated their effectiveness in learning powerful representations for complex graph datasets. However, when dealing with dynamic graphs, solely using static GNNs may not be sufficient for capturing relevant temporal patterns. Multiple research efforts have been conducted to address this limitation, primarily focusing on extending static GNNs to dynamic GNNs by incorporating a sequential model.

In [34], the authors proposed a first dynamic graph neural network model by combining a GNN with a recurrent neural network such as a vanilla RNN [27], a GRU [7] or an LSTM [16]. The authors proposed two variants. The first variant employs a combination of Convolution layers and Recurrent Neural Networks. The second variant adopts a convolutional LSTM that replaces fully connected operations in the recurrent neural network definition with convolutional operations. In [22], Kumar *et al.* proposed *JODIE*, a coupled recurrent neural network model designed to acquire trajectories of node embeddings. They employed a recurrent neural network to update a node embedding given historical information. Additionally, to capture the trajectory of future embeddings, they introduced a projection operator that can learn a node's embedding at any given point. The work in [33] calculates node representations by employing simultaneous self-attention mechanisms that consider neighborhood structural information and temporal dynamics. EvolveGCN [30] employs a graph convolution network (GCN) and a recurrent neural network. Unlike prior methods, this approach utilizes the RNN to update the GCN parameters instead of the node embeddings. The authors of [14] proposed a variational graph recurrent neural network model aimed at capturing changes in dynamic graph topology and node attributes. Additional notable examples of dynamic graph neural networks have been introduced [26], [35], [23], [19]. These techniques are primarily developed for supervised tasks such as node classification and edge prediction, but they overlook the problem of clustering in dynamic graphs.

Deep clustering has demonstrated remarkable performance in partitioning graph data. Contrastive learning has been applied within deep clustering frameworks. A node-level loss function is employed during the pretraining stage to develop a versatile node representation. Inspired by image augmentation techniques, graph augmentations like edge modification and node feature masking are employed by models to uncover deeper semantic information. This is achieved by creating more complex pretext tasks challenging the model to learn effectively. In the clustering phase, the contrastive method moves a particular node closer to its corresponding centroid and further from other centroids, thereby reshaping the representation space to make the clusters separable.

3 Methodology

We introduce MLCDG, a novel graph neural network model designed to address the problem of detecting communities in evolving graphs. We have adopted the contrastive learning paradigm as the foundation of our approach. The model architecture consists of T (the number of timesteps) Graph Neural Networks interconnected with GRU units. The t^{th} GNN takes the A^t (the t^{th} adjacency matrix) and X^t (The t^{th} attribute matrix) as input and projects the graph into a low dimensional latent space—this projection in denoted H^t. H^t and H^{t-1}

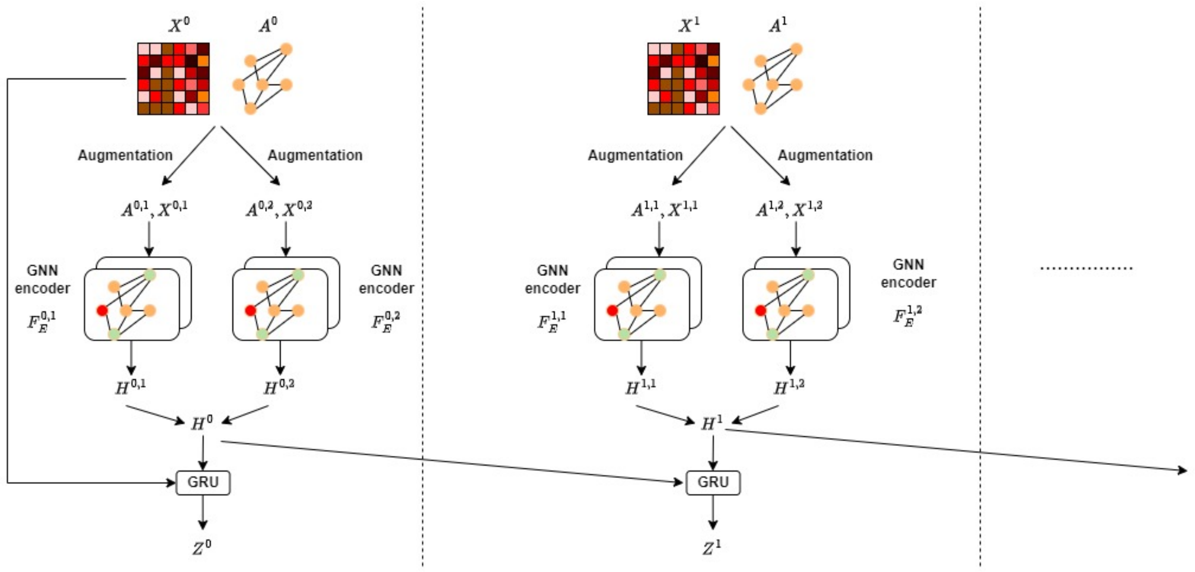

Fig. 1. The MLCDG Architecture.

represent the inputs of a GRU unit, which produces the graph's embedding at the t^{th} timestep, denoted Z^t. The architecture of MLCDG is illustrated in Fig. 1.

MLCDG is based on contrastive learning. Given the unsupervised nature of the clustering problem, creating two different graph augmentations is necessary. These augmentations or graph views are denoted $(A^{t,1})$ and $(A^{t,2})$. We denote by F_E^t (the view index is omitted to simplify notation) the GNN encoder at the t^{th} timestep and L the number of layers. We employ a Convolutional Graph Network (GCN), as described in [20].

$$H_L^t = F_E^t(A^t, X^t) = ReLU(\tilde{\Delta}^{t^{-\frac{1}{2}}} \tilde{A}^t \tilde{\Delta}^{t^{-\frac{1}{2}}} H_{L-1}^t W_L^t) \tag{1}$$

$$Z^t = GRU(H^t, H^{t-1}) \tag{2}$$

Where W_L^t represent the learning parameters of the GCN F_E^t, ReLU is the rectified linear unit activation function corresponding to the output layer of the GCN, $\tilde{\Delta}^{t^{-\frac{1}{2}}} \tilde{A}^t \tilde{\Delta}^{t^{-\frac{1}{2}}}$ represents the normalized adjacency matrix where $\tilde{A}^t = A + I$, I is the identity matrix and $\tilde{\Delta} = A\mathbf{1}_N$.

Mutual Information (MI): Mutual information measures the statistical dependence or information shared between two random variables. It quantifies how much knowing the value of one variable tells you about the other. In other words, it measures the reduction in uncertainty about one variable when you know the value of the other. Mathematically, the mutual information between two random variables X and Y is defined as:

$$I(X,Y) = \mathrm{E}_{p(x,y)}(\log(p(x,y))/p(x),p(y)) \tag{3}$$

where $p(x,y)$ represents the joint density of X and Y, $p(x)$ and $p(y)$ denote, respectively, the marginal density of X and Y.

Over the past few years, numerous research studies have demonstrated significant achievements in representation learning through the optimization of mutual information between two different data views [5], [15], [36]. Given the difficulty of estimating mutual information [32], it is expected to maximize MI by first deriving a lower bound and then optimizing this lower bound. For instance, InfoNCE [29] (Noise Contrastive Estimator) is frequently used as the lower bound of MI.

$$I(X,Y) \geq I_{NCE} = \mathrm{E}\Big(\frac{1}{M}\sum_{i=1}^{M}\frac{\exp(\theta(x_i,y_i))}{\frac{1}{M}\sum_{j=1}^{M}\exp(\theta(x_j,y_j))}\Big) \tag{4}$$

$\{x_i, y_i\}_{i=1}^{M}$ represent M independent samples from the joint density $p(x,y)$, θ is a critic function. This function aims to assign large scores to pairs (x_i, y_i) drown together (positive pairs) and small scores to negative pairs $(\{x_i, y_j\}_{i\neq j})$. The MLCGD is trained in two distinct phases: first, a pretraining phase is performed to solve a pretext task, followed by a clustering phase with the goal of refining clusters using pseudo-labels.

First phase (Pretraining): Before delving into the primary task, which is pseudo-supervision or clustering, an initial pretraining phase is conducted. The pretraining allows the model to acquire high-level representations from which pseudo-labels can be derived. At the heart of this phase, two contrastive loss functions are introduced. The first loss function is a contrastive loss applied at the node-level. We pick two graph augmentations, yielding two different views of the graph at every time step. At the time step t, the two views of the graph (A^t, X^t) are denoted $(A^{t,1}, X^{t,1})$ and $(A^{t,2}, X^{t,2})$. These augmentations are fed to two GNNs producing latent representations to the two graph views denoted $H^{t,1}$ and $H^{t,2}$. The i^{th} row in $H^{t,l}$, denoted $h_i^{t,l}$, represents the latent representation of of the node i in the graph view l at the t^{th} timestep. Given a node i, the positive samples associated to $h_i^{t,l}$ $(l \in \{1,2\}$)are $\{h_i^{t,k}\}_{k\neq l}$ and the negative samples are sampled from the set $\{h_j^{t,k}\}_{j\neq i}$. We opt for the Noise Contrastive Estimator (I_{NCE}) as our choice of a contrastive loss function. At the timestep t, the node-level contrastive loss is defined as follows:

$$\mathcal{L}_{NL} = \sum_{l=1}^{2}\sum_{i=1}^{N} -\log\Big(\frac{exp((h_i^{t,l})^T h_i^{t,k})}{\sum_{k=1}^{2}\sum_{j=1}^{N} exp((h_i^{t,l})^T h_j^{t,k})}\Big) \tag{5}$$

The critic function in (4) is a dot product: $\theta(x,y) = x^T y$. The loss L_{NL} is minimized when the representations of a specific node in both of its distinct views are embedded in such a way that they are in close proximity to each other while simultaneously being positioned far apart from the representations of other nodes. Conversely, this loss is maximized when the embeddings of the two views of the same node are situated far from each other and in close proximity to other nodes.

We leverage two graph augmentations: (i) Node Feature Masking (NFM), Which is a commonly employed attribute-based augmentation technique [17], [42]. Specifically, NFM involves the random masking of features for a subset of nodes in the provided graph. The masking can be applied either row-wise or column-wise. (ii) Graph Diffusion (GD) [9] which is a topological augmentation. The concept behind graph diffusion involves enhancing the augmented graph by incorporating global structural information by adding connections between nodes and their indirect neighbors.

In the context of evolving graphs, characterized by continuous modifications involving both the addition and removal of nodes as well as alterations in their interactions over time, our first main assumption is that these dynamic changes occur in a gradual and smooth manner, thereby avoiding sudden and abrupt shifts in the underlying community structure within the graph. To encourage the preservation of temporal smoothness and then tackle the problem of community instability, we propose a *time-level* contrastive loss function denoted $\mathcal{L}_{TL}$. Given a node i, the goal is to ensure that the latent representation h_i^t doesn't significantly diverge from h_i^{t-1}. This loss function is defined as follows:

$$\mathcal{L}_{TL} = \sum_{i=1}^{N} - \log \Big(\frac{\exp((z_i^t)^T z_i^{t-1})}{\sum_{j=1}^{N} \exp((z_i^t)^T z_j^{t-1})} \Big) \tag{6}$$

In the pretraining phase, the the self-supervision loss is calculated as a combination of node-level and time-level losses, with appropriate balancing hyperparameters.

$$\mathcal{L}_{SS} = \lambda_N \mathcal{L}_{NL} + \lambda_T \mathcal{L}_{TL} \tag{7}$$

Where λ_N and λ_T denote the balancing hyperparameters in the self-supervised loss $\mathcal{L}_{SS}$.

Second phase (Clustering): The loss terms mentioned above contrast individual entities such as node representations, enabling the learning of the community structure. Rather than focusing solely on low-level entities like nodes in contrastive learning, we will focus on higher-level entities, specifically cluster centroids. The core idea is based on the gradual transformation of the latent representational space to be well-suited for clustering. Following the pretraining phase, we employ a clustering algorithm, like K-means, on the latent codes. At the timestep t, the cluster membership matrix Φ^t is defined by:

$$\Phi_{i,j}^t = \frac{(1 + \|z_i^t - \mu_j^k\|^2)^{-1}}{\sum_{k=1}^{K} (1 + \|z_i^t - \mu_k^t\|^2)^{-1}} \tag{8}$$

Where z_i^t denotes the embedding of the node i at the timestep t, $\{\mu_k\}_{0 \leq k \leq K}$ represent the K centroids of the clusters.

Let $\mathcal{H}_\epsilon^t$ be the set of nodes assigned to their clusters with high confidence, meaning that:

$$i \in \mathcal{H}_\epsilon^t \Leftrightarrow \Phi_{i,c_{max}} \geq \epsilon$$

Where $\Phi_{i,c_{max}}$ is the highest value in the i^{th} row of Φ^t and ϵ is a confidence threshold. Let $\{\Omega_k^t\}_{0\leq k\leq K}$ be the centroids of the nods belonging to $\mathcal{H}_\epsilon^t$. We introduce a contrastive clustering loss that aims to pull the nodes assigned with high confidence to their corresponding high confident centroids, thus creating a letent space with well separated clusters. We periodically increase the hyperparameter ϵ every M iterations, aiming to refine the community structure within the latent space by enlarging the circle of high-confidence.

$$\mathcal{L}_{CL} = \sum_{i\in\mathcal{H}_\epsilon^t} -\log\left(\frac{\exp((z_i^t)^T\Omega_{k_i}^t)}{\sum_{k=1}^{K}\exp((z_i^t)^T\Omega_k^t)}\right) \tag{9}$$

$\Omega_{k_i}^t$ is the centroid corresponding to the node i. Finally the overall loss function is a weighted sum of the previously defined loss terms.

$$\mathcal{L} = \lambda_{NL}\mathcal{L}_{NL} + \lambda_{TL}\mathcal{L}_{TL} + \lambda_{CL}\mathcal{L}_{CL} \tag{10}$$

Where λ_{NL}, λ_{TL} and λ_{CL} denote the balancing hyperparameters in the overall loss $\mathcal{L}$.

Training algorithm: Our model is pretrained for I_1 iterations. During this phase, the parameters of the graph neural networks, as well as the GRU units, are updated to minimize the loss function $\mathcal{L}_{SS}$ using the Adam optimizer. Subsequently, the clustering is performed on the latent codes using K-means. Afterward, the model is fine-tuned by minimizing the loss function $\mathcal{L}$. Algorithm 1 provides the details of the clustering phase of MLCDG.

4 Experiments

Baselines: We assess the effectiveness of our approach by conducting a comprehensive comparison with a range of methodologies. Specifically, our evaluation includes traditional clustering algorithms, deep graph clustering techniques, and dynamic graph representation learning methods, which serve as the baseline for our analysis.

- **Traditional Clustering methods**: We opted for considering two prominent traditional clustering algorithms: K-means and spectral clustering. The K-means algorithm typically operates under the assumption that clusters exhibit a spherical or concave shape, meaning data points are expected to fall within a certain radius around the cluster centroid. On the contrary, spectral clustering proves beneficial when dealing with datasets characterized by non-linear and complex cluster structures.
- **Deep Graph Clustering**: We evaluate our approach by comparing it to three deep graph clustering methods. The first model in this comparison is a graph autoencoder (**GAE**) [21], which first learns node embeddings and then employs a clustering algorithm. The second model is a variational variant of GAE(**VGAE**)[21]. **DAEGC** [38] is an other deep graph clustering approach. DAEGC utilizes an attention network to capture the significance of adjacent nodes with respect to a central target node.

Algorithm 1 MLCDG Clustering

```
**Input:** Dynamic graph $\{(A^t, X^t)\}_{0 \le t \le T-1}$; number of clusters $K$; threshold $\epsilon$; number of epochs $N_{epochs}$; update interval $M$; initial cluster membership matrix $\{\Phi^t\}_{0 \le t \le T-1}$
**Output:** Final cluster membership matrix $\{\Phi^t\}_{0 \le t \le T-1}$
$epoch \leftarrow 0$
for $0 \le t \le T-1$ do
    Compute two graph views $(A^{t,1}, X^{t,1})$ and $(A^{t,2}, X^{t,2})$
    for $epoch \le N_{epochs}$ do
        if $epoch \% M = 0$ then
            Update $\epsilon$
        end if
        $H^{t,1} \leftarrow F_E^{t,1}(A^{t,1}, X^{t,1})$
        $H^{t,2} \leftarrow F_E^{t,2}(A^{t,2}, X^{t,2})$
        $H^t = (H^{t,1} + H^{t,2})/2$
        $Z^t = GRU(H^t, H^{t-1})$
        Compute $\mathcal{H}_\epsilon^t$
        Compute and backpropagate $\mathcal{L}$
    end for
    Compute and store $\Phi^t$ using $Z^t$
end for
return $\{\Phi^t\}_{0 \le t \le T-1}$
```

– **Dynamic Graph Neural Networks Models**: As far as we know, our method represents the first deep graph clustering approach designed for dynamic graphs. Previous dynamic graph neural network models were primarily oriented toward different downstream tasks such as node classification, edge detection, and edge prediction. We have selected several notable examples of these prior methods as baselines for our approach. **DynGEM** [12] employs autoencoders to progressively produce node embeddings at time t, utilizing the graph snapshot from time $t-1$. **DynAERNN** [11] employs a deep architecture comprising dense and recurrent layers to capture relevant knowledge from the temporal transitions within the network. **EvolveGCN** [30] leverages both a graph convolution network and a recurrent neural network to update the learnable parameters of the GCNs over time. **VGRNN** [14] is a variational graph recurrent neural network that enhances the GRNN model by introducing high-level latent random variables into its architecture.

Datasets: Due to the scarcity of labeled dynamic graph datasets, our experiments were carried out using two distinct datasets. We collected one real-world dataset and generated one synthetic dynamic graph dataset. Despite the initial challenge posed by the limited availability of datasets, the positive outcomes of our preliminary experiments serve as a strong incentive for us to incorporate additional datasets into our future research.

– **DBLP-T** [41]: This dataset is obtained from the DBLP website, which offers accessible bibliographic datasets for prominent computer science journals

	Baseline	DBLP-T		SynSBM	
		ACC	F1-Sc	ACC	F1-Sc
Classical Methods	K-Means	58.83	<u>75.22</u>	41.30	49.02
	Spectral Clustering	58.93	73.22	26.00	23.66
Static Deep Graph Clustering	GAE	57.87	74.55	45.69	53.21
	VGAE	<u>62.76</u>	75.10	75.12	87.32
	DAEGC	59.46	73.82	81.43	84.47
Dynamic Graph Models	DynGEM	61.63	74.88	93.12	94.21
	DynAERNN	60.73	73.20	90.32	92.75
	EvolveGCN	61.02	74.52	<u>95.75</u>	<u>96.53</u>
	VGRNN	59.17	73.58	91.13	93.79
	MLCDG	**72.31**	**79.01**	**98.10**	**98.10**

Table 1. Clustering results on two dynamic graph datasets. Best methods in bold and second best underlined.

and conferences. The authors are represented by nodes and the co-authorship by edges. Each year corresponds to a single time step, with co-author connections being established in the year a collaborative paper is published. The two classes represent the research domain (*computer networks* or *machine learning*). The node class has the potential to undergo changes over time.

– **SynSBM**: We slightly modify the static stochastic block (SBM) model to generate dynamic graphs exhibiting community structure. After generating an initial graph using a static SBM, a number of nodes are randomly selected and assigned to other communities. The connections of these nodes are added according to the two parameters of the SBM: α and β represent, respectively, the probabilities of a node making connections within its own community and with nodes in other communities.

Hypermapameters: In the MLCDG algorithm, we set the number of epochs to 500 during the second phase. Additionally, we incorporate a gradual increase in ϵ value, starting from 0.4 and incrementing by 0.1 every 100 epochs until it reaches 0.8.

We assess our model's performance using two commonly employed metrics: Accuracy (ACC) and the F1-score (F1-Sc). Let y_{true} be the true labels vector and y_{pred} denote the predicted labels. Table 1 displays the preliminary results of our model alongside the results from the baseline models. The results represent the mean performance metrics across all snapshots. Table 1 shows that MLCDG surpasses all comparing models, notably achieving a remarkable improvement of 13% in ACC and 5% in F1-Sc when compared to the top-performing baseline on the DBLP-T dataset. Regarding the SynSBM synthetic dataset, our model maintains its superiority by achieving an increase of 2% and 1.6% in ACC and F1-Sc, respectively, when compared to the top-performing baseline. The lower performance of the baselines can be attributed to the inability of static and dynamic graph neural networks to effectively capture temporal dependencies using basic graph and feature reconstruction losses. This leads to a failure in learning latent representations suitable for clustering. Furthermore, we observed a consistent improvement in clustering performance as time advances, indicating that our approach benefits from the communities identified earlier to discover new communities effectively.

We argue that the effectiveness of our model stems from two key factors. First, the combination of a recurrent neural network and the temporal loss allowed our model to capture relevant historical information and mitigate the negative impact of the well-known problem of community instability. Second, by incorporating the proposed clustering term into the loss function, which relies on highly confident clustering assignments, the latent representational space can be gradually reshaped to enhance the separability of the clusters.

5 Conclusion

In this paper, we presented MLCDG, a novel deep clustering framework for dynamic attributed graphs, leveraging multi-level contrastive learning to address the challenges of community detection in evolving networks. Our method integrates node-level, temporal-level, and cluster-level contrastive losses to foster stable and meaningful clustering over time. By employing a Dynamic Graph Neural Network architecture, MLCDG effectively captures the evolving nature of graph data, maintaining temporal coherence and enhancing the quality of community detection. By combining the strengths of contrastive learning and deep graph neural networks, MLCDG represents an advancement in dynamic graph clustering. It opens up new possibilities for understanding complex systems in which the underlying graph data evolve over time. In line with this perspective, we are collecting additional real datasets to further investigate our methodology under various real-world scenarios. In addition, we plan to develop an analytical tool to track and predict structural changes and explain the evolution of communities over time. Our goal is to track and predict the evolution of community structures and the variation of the role of nodes within communities over short and long-term scales.

References

1. Ahmadian, S., Joorabloo, N., Jalili, M., Meghdadi, M., Afsharchi, M., Ren, Y.: A temporal clustering approach for social recommender systems. In: 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM). pp. 1139–1144. IEEE (2018)
2. Aynaud, T., Guillaume, J.L.: Static community detection algorithms for evolving networks. In: 8th international symposium on modeling and optimization in mobile, ad hoc, and wireless networks. pp. 513–519. IEEE (2010)
3. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of statistical mechanics: theory and experiment **2008**(10), P10008 (2008)

4. Bourqui, R., Gilbert, F., Simonetto, P., Zaidi, F., Sharan, U., Jourdan, F.: Detecting structural changes and command hierarchies in dynamic social networks. In: 2009 International conference on advances in social network analysis and mining. pp. 83–88. IEEE (2009)
5. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: International conference on machine learning. pp. 1597–1607. PMLR (2020)
6. Chen, Z., Wilson, K.A., Jin, Y., Hendrix, W., Samatova, N.F.: Detecting and tracking community dynamics in evolutionary networks. In: 2010 IEEE International Conference on Data Mining Workshops. pp. 318–327. IEEE (2010)
7. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 (2014)
8. Folino, F., Pizzuti, C.: An evolutionary multiobjective approach for community discovery in dynamic networks. IEEE Transactions on Knowledge and Data Engineering **26**(8), 1838–1852 (2013)
9. Gasteiger, J., Weißenberger, S., Günnemann, S.: Diffusion improves graph learning. Advances in neural information processing systems **32** (2019)
10. Girvan, M., Newman, M.E.: Community structure in social and biological networks. Proceedings of the national academy of sciences **99**(12), 7821–7826 (2002)
11. Goyal, P., Chhetri, S.R., Canedo, A.: dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. Knowledge-Based Systems **187**, 104816 (2020)
12. Goyal, P., Kamra, N., He, X., Liu, Y.: Dyngem: Deep embedding method for dynamic graphs. arXiv preprint arXiv:1805.11273 (2018)
13. Guo, X., Gao, L., Liu, X., Yin, J.: Improved deep embedded clustering with local structure preservation. In: Ijcai. vol. 17, pp. 1753–1759 (2017)
14. Hajiramezanali, E., Hasanzadeh, A., Narayanan, K., Duffield, N., Zhou, M., Qian, X.: Variational graph recurrent neural networks. Advances in neural information processing systems **32** (2019)
15. Hjelm, R.D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., Bengio, Y.: Learning deep representations by mutual information estimation and maximization. arXiv preprint arXiv:1808.06670 (2018)
16. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
17. Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., Leskovec, J.: Strategies for pre-training graph neural networks. arXiv preprint arXiv:1905.12265 (2019)
18. Hui, B., Zhu, P., Hu, Q.: Collaborative graph convolutional networks: Unsupervised learning meets semi-supervised learning. In: Proceedings of the AAAI conference on artificial intelligence. vol. 34, pp. 4215–4222 (2020)
19. Jin, W., Qu, M., Jin, X., Ren, X.: Recurrent event network: Autoregressive structure inference over temporal knowledge graphs. arXiv preprint arXiv:1904.05530 (2019)
20. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
21. Kipf, T.N., Welling, M.: Variational graph auto-encoders. arXiv preprint arXiv:1611.07308 (2016)
22. Kumar, S., Zhang, X., Leskovec, J.: Predicting dynamic embedding trajectory in temporal interaction networks. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. pp. 1269–1278 (2019)

23. Li, J., Han, Z., Cheng, H., Su, J., Wang, P., Zhang, J., Pan, L.: Predicting path failure in time-evolving graphs. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. pp. 1279–1289 (2019)
24. Lin, Y.R., Chi, Y., Zhu, S., Sundaram, H., Tseng, B.L.: Facetnet: a framework for analyzing communities and their evolutions in dynamic networks. In: Proceedings of the 17th international conference on World Wide Web. pp. 685–694 (2008)
25. Liu, Y., Jin, M., Pan, S., Zhou, C., Zheng, Y., Xia, F., Philip, S.Y.: Graph self-supervised learning: A survey. IEEE Transactions on Knowledge and Data Engineering **35**(6), 5879–5900 (2022)
26. Manessi, F., Rozza, A., Manzo, M.: Dynamic graph convolutional networks. Pattern Recognition **97**, 107000 (2020)
27. Medsker, L.R., Jain, L.: Recurrent neural networks. Design and Applications **5**(64-67), 2 (2001)
28. Mrabah, N., Bouguessa, M., Touati, M.F., Ksantini, R.: Rethinking graph autoencoder models for attributed graph clustering. IEEE Transactions on Knowledge and Data Engineering (2022)
29. Oord, A.v.d., Li, Y., Vinyals, O.: Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748 (2018)
30. Pareja, A., Domeniconi, G., Chen, J., Ma, T., Suzumura, T., Kanezashi, H., Kaler, T., Schardl, T., Leiserson, C.: Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In: Proceedings of the AAAI conference on artificial intelligence. vol. 34, pp. 5363–5370 (2020)
31. Pizzuti, C.: Ga-net: A genetic algorithm for community detection in social networks. In: International conference on parallel problem solving from nature. pp. 1081–1090. Springer (2008)
32. Poole, B., Ozair, S., Van Den Oord, A., Alemi, A., Tucker, G.: On variational bounds of mutual information. In: International Conference on Machine Learning. pp. 5171–5180. PMLR (2019)
33. Sankar, A., Wu, Y., Gou, L., Zhang, W., Yang, H.: 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. pp. 519–527 (2020)
34. Seo, Y., Defferrard, M., Vandergheynst, P., Bresson, X.: Structured sequence modeling with graph convolutional recurrent networks. In: Neural Information Processing: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13-16, 2018, Proceedings, Part I 25. pp. 362–373. Springer (2018)
35. Taheri, A., Gimpel, K., Berger-Wolf, T.: Learning to represent the evolution of dynamic graphs with recurrent models. In: Companion proceedings of the 2019 world wide web conference. pp. 301–307 (2019)
36. Veličković, P., Fedus, W., Hamilton, W.L., Liò, P., Bengio, Y., Hjelm, R.D.: Deep graph infomax. arXiv preprint arXiv:1809.10341 (2018)
37. Vignac, C., Loukas, A., Frossard, P.: Building powerful and equivariant graph neural networks with structural message-passing. Advances in neural information processing systems **33**, 14143–14155 (2020)
38. Wang, C., Pan, S., Hu, R., Long, G., Jiang, J., Zhang, C.: Attributed graph clustering: A deep attentional embedding approach. arXiv preprint arXiv:1906.06532 (2019)
39. Wang, Y., Wu, B., Pei, X.: Commtracker: A core-based algorithm of tracking community evolution. In: Advanced Data Mining and Applications: 4th International Conference, ADMA 2008, Chengdu, China, October 8-10, 2008. Proceedings 4. pp. 229–240. Springer (2008)

40. Xie, J., Girshick, R., Farhadi, A.: Unsupervised deep embedding for clustering analysis. In: International conference on machine learning. pp. 478–487. PMLR (2016)
41. Yao, Y., Joe-Wong, C.: Interpretable clustering on dynamic graphs with recurrent graph neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 35, pp. 4608–4616 (2021)
42. Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., Wang, L.: Deep graph contrastive representation learning. arXiv preprint arXiv:2006.04131 (2020)