Deep Community Detection in Attributed Temporal Graphs: Experimental Evaluation of Current Approaches

Nelson A. R. A. Passos
University of Pisa
Pisa, Italy
National Research Council
Pisa, Italy
nelson.reis@phd.unipi.it

Emanuele Carlini
National Research Council
Pisa, Italy
emanuele.carlini@isti.cnr.it

Salvatore Trani
National Research Council
Pisa, Italy
salvatore.trani@isti.cnr.it

Abstract

Recent advances in network representation learning have sparked renewed interest in developing strategies for learning on spatio-temporal signals, crucial for applications like traffic forecasting, recommendation systems, and social network analysis. Despite the popularity of Graph Neural Networks for node-level clustering, most specialized solutions are evaluated in transductive learning settings, where the entire graph is available during training, leaving a significant gap in understanding their performance in inductive learning settings. This work presents an experimental evaluation of community detection approaches on temporal graphs, comparing traditional methods with deep learning models geared toward node-level clustering. We assess their performance on six real-world datasets, focused on a transductive setting and extending to an inductive setting for one dataset. Our results show that deep learning models for graphs do not consistently outperform more established methods on this task, highlighting the need for more effective approaches and comprehensive benchmarks for their evaluation.

CCS Concepts

• **Mathematics of computing** → **Graph theory**; • **Computer systems organization** → **Neural networks**; • **Information systems** → **Clustering and classification**; • **Computing methodologies** → **Unsupervised learning**.

Keywords

Graph Neural Networks, Node Clustering, Temporal Graphs.

ACM Reference Format:
Nelson A. R. A. Passos, Emanuele Carlini, and Salvatore Trani. 2024. Deep Community Detection in Attributed Temporal Graphs: Experimental Evaluation of Current Approaches. In *Proceedings of the 3rd GNNet Workshop: Graph Neural Networking Workshop (GNNet '24), December 9–12, 2024, Los Angeles, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3694811.3697822

1 Introduction

The task of dividing a network into groups is crucial for understanding its structure and dynamics and is a well-defined research topic in Network Science [28]. It has also gained traction in the Machine Learning community in recent years, which has ensued an exploration of the use of encoders and deep learning models for network representation, such as Graph Neural Networks (GNNs). However, even though GNNs may be considered to constitute the state of the art for node-level classification and clustering tasks [37], there is still comparatively little research and a lack of rigorous evaluation on employing them on dynamic graphs [22] — in which nodes, edges, and features may altogether change over time. Real-world networks are generally dynamic in nature, but most of the existing research on partitioning them has traditionally focused on static graphs. Moreover, the majority of studies carrying mesoscale network analyses still do not employ algorithms or models designed for temporal graphs, relying instead on their static counterparts, due to widespread availability and lowered computational costs.

This paper aims to shed some light on the matter, by evaluating the performance of distinct approaches for clustering nodes. Our experiment considers six real-world graph datasets from different domains and scales, and eight different baselines for performance comparison. We include established methods for community detection, network representation, and deep learning on graphs, the latter geared specifically toward the task of node clustering, i.e., "deep" community detection. A pretraining stage is initially employed to obtain node features by random walk sampling, followed by training with models that consider the learned embeddings and the graph's topology. We further extend our evaluation to inductive learning on a single attributed temporal graph with readily available node-level features, to verify the models' generalization power on unseen nodes. Although we focus on the task of node-level clustering, we highlight that similar models may potentially be used to other downstream tasks, e.g., graph classification, especially those that may benefit from learning the network's evolution over time.

Our contributions therefore are the following:

- We perform a transductive learning evaluation of eight distinct approaches for node-level clustering on graphs, considering six real-world datasets of different scales and domains.
- We extend our evaluation when possible to an inductive learning setting, with the goal of understanding how they perform in terms of generalization power on unseen data.
- We release a dynamic graph dataset based on PubMed [3], a popular citation network primarily used in node classification tasks, now available with edge-level temporal data.

This work is structured as follows. **Section 2** offers a brief overview of related work, especially on deep learning models for

GNNet '24, December 9–12, 2024, Los Angeles, CA, USA

ACM ISBN 979-8-4007-1254-8/24/12
https://doi.org/10.1145/3694811.3697822

static and temporal graphs. **Section 3** formalizes the research problem, presents the models and baselines considered, and describes the datasets and metrics used in our experiments. **Section 4** discusses our experimental results based on the models' performance. Lastly, **Section 5** offers some closing thoughts and outlines the research directions we aim to explore in the near future.

2 Related Work

In the context of networks, a cluster, module, or community is broadly defined as a group of objects densely connected among themselves, but sparsely connected to other groups [28]. The most widely adopted methods for their detection nowadays rely on agglomerative strategies that optimize some objective function to maximize the quality of the partitioning, of which the most common is modularity [36]; statistical inference, which estimate the data's likelihood through probabilistic generative (e.g., Bayesian) processes, such as the Stochastic Block Model [31]; network representation learning techniques, that map nodes into a real dimensional space by employing some sort of encoder, mostly focused on the Skip-Gram model [25]; and deep learning models designed to operate directly on relational data, such as Graph Neural Networks (GNNs), discussed in more detail next. Other methods not covered in this work include divisive strategies; statistical physics models; matrix factorization techniques; information-theoretic approaches; label and belief propagation; and other additional, mixed strategies. We direct the reader to [5] for a review on some of these methods.

Deep learning on graphs. Most recent advances on graph representation have been achieved by employing deep learning models that operate directly on graphs. In a paradigm known as message passing [8], GNNs obtain node embeddings $\mathbf{H}$ by recursively combining invariant to permutation their and their neighbors' features, i.e., $h_i = \phi\left(x_i, \oplus_{j \in \mathcal{N}_i} \psi(x_i, x_j, x_e)\right)$, where $\mathcal{N}$ are neighbors, $\oplus$ is an aggregation function, and ϕ and ψ are differentiable functions that learn representations from node and edge features x_i, x_j, x_e.

The learning process in this framework is usually based on gradient descent, in which weights are updated according to the derivatives computed in the last step, while the loss is minimized until a predetermined step or an approximate fixed point solution. Employing multiple layers allows considering higher-order information beyond the node's immediate neighborhood, and the feature maps may be downsampled to reduce their computational cost. Lastly, local (node-level) or global (graph-level) pooling operations may be used to select, reduce, and connect [9] or readout embeddings with permutation-equivariant or permutation-invariant functions — allowing them to be used for a variety of downstream tasks, such as node clustering, link prediction, and graph classification.

Many models following this paradigm have since been proposed, extending it to employ more sophisticated convolutional operations, based on the graph's spectral or spatial properties [14, 17] properties; adapting autoencoders [16] to graphs, in which a decoder and an encoder function are used to minimize the loss of reconstructing their topology into the original form; introducing the use of attention mechanisms [1, 38] that weigh the importance of nodes and features differently according to some criteria; considering $k-$tuples of nodes (subgraphs) for higher-order message passing [26]; or by considering the graph's temporal dynamics, which we further elaborate next. For a comprehensive introduction to GNNs and a survey covering the models mentioned herein, see [19, 41].

Deep learning on temporal graphs. GNNs for dynamic graphs rely on the same paradigm as static models to learn on non-Euclidean data, but extend them to consider the temporal dimension of the graph, either by taking snapshots [29] or sequences of events [11, 33] as input. In general, recurrent mechanisms are used to progressively learn on spatio-temporal signals: such as Long Short-Term Memory (LSTM) [6], in which cell states contain three gates that control the amount of information to be discarded (forget gate), added (input gate), and transferred (output gate) to the next layer, or Gated Recurrent Units (GRUs) [2], a simplification of LSTM that maintains a hidden state only, controlled by a reset and an update gate. Alternatively, models may employ variational autoencoders [12], attention-wise [34, 42], or temporal decay functions [21] instead — such as the Hawkes process [15], in which past events have a (usually exponentially) decreasing effect on future probabilities, conditioned by the intensity of previously observed events.

Similarly to static GNNs, however, the design of temporal GNNs is mostly based on empirical intuition, heuristics, and experimental trial and error [43], and there is little theoretical understanding of their limitations beyond the sensible statement that their expressiveness is likely limited in the same regards [26]. Therefore, despite the recent advances in the field and its potential relevance in diverse application domains, the task of node classification or clustering of dynamic graphs is still an open research question, and the evaluation of their performance on node-level clustering an interesting and relevant objective. For a recent survey on temporal GNNs, including a proposed taxonomy and a discussion on their limitations and research opportunities, we direct the reader to [22].

3 Methodology

3.1 Research Question

Our research question concerns the unsupervised node-level clustering of attributed temporal graphs: *how efficiently the current state-of-the-art models learn node representations of a temporal graph and partition them into groups?* In this work, we compare different approaches to better understand how well they perform in this task, compared to distinct commonly employed methods as baselines.

A temporal graph can be defined as $\mathcal{G} \coloneqq \{\mathcal{V}, \mathcal{E}, X_{\mathcal{V}}, X_{\mathcal{E}}\}$, in which $\mathcal{V}$ and $\mathcal{E}$ are the sets of all temporal nodes and edges, and $X_{\mathcal{V}} \in \mathbb{R}$ and $X_{\mathcal{E}} \in \mathbb{R}$ are (optional) node and edge-level attribute features. A temporal graph may be computationally represented in different ways, leading to distinct algorithmic solutions [22]. For instance, it may be represented as a sequence of snapshots, i.e., $\mathcal{G}_S \coloneqq \{G_1, \ldots, G_t \mid t \in \mathbb{N}\}$, where $G_t \coloneqq \{V, E, X_V, X_E\}$ is a static graph at time t; or alternatively, as a sequence of edge-level events, i.e., $\mathcal{G}_E \coloneqq \{\forall e \in \mathcal{E} : \{u, v, t, \delta, x_u, x_v, x_e\} \mid t \in \mathbb{R}^+\}$, where u and v are temporal nodes, t is the time of the event, $\delta \in \{0, 1\}$ constitutes an edge addition or removal, and x_u, x_v, x_e are node-level and edge-level features. Note that snapshot-based and event-based temporal graphs are also found in the literature as discrete-time temporal graphs (DTTG) and continuous-time temporal graphs (CTTG).

The goal of a model (algorithm) is therefore learning (employing) a function that efficiently maps nodes into groups, that is,

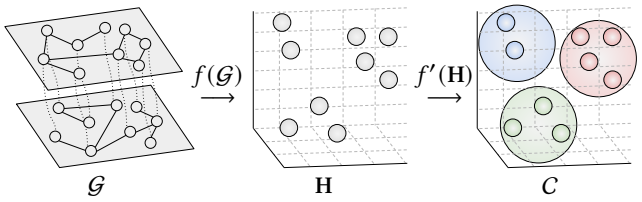

Figure 1: Clustering node representations. A function f is applied to a temporal graph $\mathcal{G}$ to learn node embeddings $\mathbf{H} \in \mathbb{R}^d$, here illustrated as snapshots and a $d = 3$ dimensional space, respectively. A second function f' obtains the resulting node clusters C from it.

$f : \mathcal{G} \rightarrow C \coloneqq \{ C_n \subseteq \mathcal{V} \mid n \geq 1 \}$, in which C is the set of all clusters (communities) in the graph. It may consider the graph's topology, node and edge features, and temporal dynamics in order to produce subsets of nodes that best represents its underlying mesoscale structures. In the network representation learning paradigm, commonly two functions are employed instead: the first encodes nodes into a real $d-$dimensional space, and the second divides them into groups based on their latent representations in an unsupervised manner (e.g., K-Means[1]), as illustrated in Figure 1. Alternatively, GNN-based models may also employ a pooling strategy in order to obtain the final clusters from the learned embeddings.

3.2 Models

We have selected eight distinct models for comparison. Our focus was primarily on their relevance, widespread adoption, presence as baselines in the literature, and implementation availability using open source frameworks [4, 30]. We compare some of their characteristics in Table 1 and briefly describe each one next.

Model	Input	Topology	Features	Temporal
K-Means	X_V		✓	
Spectral Clustering	G	✓		
Leiden	G	✓		
Node2Vec	G	✓		
DynNode2Vec	$\mathcal{G}_S$	✓		✓
tNodeEmbed	$\mathcal{G}_S$	✓		✓
DAEGC	G	✓	✓	
TGC	$\mathcal{G}_E$	✓	✓	✓

Table 1: Models. Table divided into three groups: general algorithms, shallow encoders for graph learning, and GNN models. Input: X_V, G, $\mathcal{G}_S$ and $\mathcal{G}_E$ are node-level features, static graphs, snapshot-based and event-based temporal graphs, respectively.

K-Means[2] is a clustering algorithm that partitions elements into k clusters based on their positions in the feature space. **Spectral Clustering** is an algorithm to partition nodes into k clusters by eigendecomposition of the graph's Laplacian. **Leiden** [36] is an optimization algorithm based on the graph's topology; we employ modularity as a quality function. **Node2Vec** [10] is a network representation learning algorithm that employs a shallow encoder to map node similarities based on their co-occurrences in random walks. **DynNode2Vec** [23] is an extension of Node2Vec that samples a sequence of temporal graph snapshots, with initial weights defined by the previously learned embeddings. **tNodeEmbed** [35] similarly considers graph snapshots, but employs matrix approximations to sequentially align their node embeddings. **DAEGC** [40] is a static graph autoencoder that employs a 2-layer graph attentional network to learn node embeddings, based on their first- and second-order neighborhoods, and minimizes a joint reconstruction and clustering loss. **TGC** [21] is a temporal graph autoencoder that employs the Hawkes function to learn node embeddings, in which the importance of their features decay as a function of time elapsed.

As far as we are aware, TGC is the only GNN model so far released that is specifically designed for the task of node clustering in attributed temporal graphs [21, 22]. While it has recently been shown to outperform most baselines, we also chose to include DAEGC for being an autoencoder-based architecture that is also geared toward node clustering in attributed static graphs, so to allow comparing the former with a similar approach, but which does not exploit a network's temporal dynamics during learning.

3.3 Datasets

We have considered six publicly available real-world networks to evaluate the models' performance on node-level clustering tasks.

Dataset	$\lvert V \rvert$	$\lvert E \rvert$	$\lvert \mathcal{E} \rvert$	S	d^V	y	t
arXivAI	69 854	696 819	696 819	244	128	5	27
Brain	5 000	883 207	1 007 744	1	128	10	12
DBLP	28 085	150 571	222 169	113	128	10	27
Patent	12 214	41 916	41 916	5	128	6	891
PubMed★	19 717	44 324	44 324	1	500	3	42
School	327	5 818	188 508	1	128	9	7 375

Table 2: Datasets. $|V|$ nodes, $|E|$ edges, $|\mathcal{E}|$ temporal edges, S subgraphs (connected components), d^V-dimensional node features, y classes, and t time steps. A star (★) marks datasets with original node features, otherwise obtained by pretraining with Node2Vec.

We display their main characteristics in Table 2. **arXivAI** [20] is a citation network of papers in Artificial Intelligence. **Brain** [32] is a network of human brain regions and their functional connectivity. **DBLP** [45] is a co-author network in Computer Science [20]. **Patent** [13] is a network of patents from the US Patent Office. **PubMed** [3] is a citation network of papers in Medicine, with most relevant terms in their abstracts included as node-level features. We queried the official API to obtain the publication dates of the papers by their IDs, and added it as an edge-level attribute considering yearly intervals — resulting in a graph of the same order, size, and feature dimensionality as in [44], itself based on [27]. Lastly, **School** [24] is a network of interactions between students in a high school. All

[1]For evaluation purposes, it is common to use K-Means to compare the separability of high-dimensional node embeddings generated by distinct models. In other application scenarios, more refined functions may be employed instead, allowing to fulfill different requirements — such as that of producing overlapping (mixed membership) clusters.
[2]Note that we employed K-Means solely as a baseline to allow comparing the separability of node embeddings obtained by each model — except in cases when clustering is performed directly on the graph, instead of its learned node-level representations.

datasets are publicly available and used in the literature to evaluate the performance of models on node-level prediction tasks [21].

Following the same procedure as [21], node features for the unattributed graphs (all in Table 2, except PubMed) were generated with Node2Vec [10], with p and q set to 1 and the number of walks and walk length set to 10 and 80, respectively. All graphs were first preprocessed to remove isolated nodes and self-loops, and the node embeddings were normalized to have zero mean and unit variance. The resulting datasets were used to evaluate the models in a transductive learning setting, in which the whole graph is available during message passing and the learned representations are used to predict the labels of the nodes in the test set only.

In contrast, our evaluation considering an inductive learning setting was herein restricted to PubMed, being the only dataset with "original", i.e., readily available node-level features. Note that this limitation is due to avoid the possibility of information leakage during learning, as considering node-level features coming from, e.g., Node2Vec would imply instead that nodes supposed to be unseen during training might be sampled during the random walks.

3.4 Evaluation Metrics

The following metrics were chosen to evaluate the performance of the selected models, guided by their robustness for clustering tasks:

- **ACC**: accuracy or proportion of correctly classified nodes; we use the Kuhn-Munkres algorithm for label assignments.
- **AMI** [39]: Adjusted Mutual Information, based on the mutual information between true and predicted clusters. In contrast to its normalized version, this metric is adjusted by chance.
- **ARI** [39]: Adjusted Rand Index, based on the number of pairs of elements assigned to the same or distinct groups.

The three metrics are commonly used in the literature to evaluate multi-class clustering tasks and allow their systematic comparison. We note that we have omitted the commonly employed F1-score due to space constraints and a high correlation (0.96) with ACC.

4 Experimental Results

We present our experimental results for the models, datasets and metrics described in previous sections. Each model was run 5 times with fixed seeds to account for randomness and allow reproducible results. Table 3 reports their average performance in a transductive learning setting, and Table 4 in the inductive learning setting. All models were trained on a single NVIDIA A100 GPU (80 GB).

In a **transductive learning setting**, considering the metrics adjusted for chance and their reported uncertainty, we observe that the GNN-based solutions we experimented with — TGC (temporal) and DAEGC (static) — consistently yielded the best performance results for only one out of six datasets we selected for evaluation, i.e., *DBLP*. On the *PubMed* dataset, DAEGC reported the highest mean, but also the highest variability among all models and metrics. In all other cases, both were either outperformed (*Brain*, *Patent*) or yielded comparable results (*arXivAI*, *School*) to baselines, which implies that current GNNs may still underperform in unsupervised node clustering tasks when compared to more established techniques — especially in the case of unattributed graphs, in which their expressiveness is limited by the lack of node-level features.

Although a pretraining stage with Node2Vec was employed to add features to nodes in unattributed graphs, i.e., all but *PubMed*, the GNNs' expressiveness may still have been hindered by the fact that their features were initially generated by exploiting only the graph's topology during walk sampling — which, in turn, may not have been able to capture well the underlying mesoscale structures of more complex networks. For example, on the larger-scale *arXivAI*

Dataset	Model	ACC	AMI	ARI
arXivAI	Spectral	.389 ± .002	.016 ± .008	.012 ± .006
	Leiden	.525 ± .033	.302 ± .027	.239 ± .031
	Node2Vec	*.646 ± .001*	**.363 ± .001**	**.404 ± .001**
	DynNode2Vec	.268 ± .001	.001 ± .001	.000 ± .001
	tNodeEmbed	**.673 ± .001**	*.299 ± .001*	*.312 ± .001*
	DAEGC	OOM	OOM	OOM
	TGC	*.646 ± .001*	**.363 ± .001**	**.404 ± .001**
Brain	Spectral	**.485 ± .001**	**.498 ± .001**	**.320 ± .001**
	Leiden	.404 ± .011	.470 ± .016	.300 ± .017
	Node2Vec	*.452 ± .002*	.466 ± .001	.270 ± .001
	DynNode2Vec	.163 ± .002	.049 ± .001	.019 ± .001
	tNodeEmbed	.417 ± .002	.434 ± .001	.243 ± .002
	DAEGC	.414 ± .010	.431 ± .009	.253 ± .010
	TGC	.434 ± .001	*.497 ± .003*	*.288 ± .003*
DBLP	Spectral	.289 ± .001	.008 ± .001	.000 ± .001
	Leiden	.400 ± .013	.302 ± .003	.187 ± .005
	Node2Vec	*.466 ± .001*	*.351 ± .001*	.207 ± .001
	DynNode2Vec	.155 ± .002	.006 ± .001	.002 ± .001
	tNodeEmbed	.451 ± .001	.345 ± .001	.203 ± .001
	DAEGC	.465 ± .010	.344 ± .004	**.219 ± .004**
	TGC	**.471 ± .001**	**.355 ± .001**	*.209 ± .001*
Patent	Spectral	**.575 ± .011**	**.365 ± .023**	**.319 ± .046**
	Leiden	.431 ± .026	.203 ± .016	.140 ± .027
	Node2Vec	.400 ± .020	.270 ± .022	.171 ± .026
	DynNode2Vec	.354 ± .038	.139 ± .045	.089 ± .042
	tNodeEmbed	.424 ± .048	.248 ± .024	.177 ± .035
	DAEGC	.462 ± .029	.315 ± .052	.271 ± .052
	TGC	*.503 ± .019*	*.329 ± .027*	*.272 ± .035*
PubMed⋆	Spectral	.593 ± .003	.162 ± .008	.143 ± .004
	K-Means	.595 ± .001	**.312 ± .001**	.281 ± .001
	Leiden	.633 ± .018	.256 ± .020	.245 ± .033
	Node2Vec	**.687 ± .001**	.289 ± .001	**.312 ± .001**
	DynNode2Vec	.543 ± .001	.192 ± .001	.135 ± .001
	tNodeEmbed	.673 ± .001	*.297 ± .001*	*.307 ± .001*
	DAEGC	.713 ± .040	.320 ± .042	.339 ± .065
	TGC	*.677 ± .001*	.254 ± .002	.280 ± .002
School	Spectral	.967 ± .001	.940 ± .001	.933 ± .000
	Leiden	.851 ± .023	.911 ± .008	.843 ± .016
	Node2Vec	**.999 ± .002**	**.998 ± .004**	**.998 ± .004**
	DynNode2Vec	.200 ± .004	.025 ± .002	.013 ± .001
	tNodeEmbed	.200 ± .002	.025 ± .001	.013 ± .001
	DAEGC	.997 ± .006	.994 ± .009	.993 ± .010
	TGC	*.997 ± .001*	*.994 ± .001*	*.994 ± .001*

Table 3: Transductive evaluation. We mark the best result for each dataset considering their uncertainty in bold, the second best in italic, and out-of-memory errors as OOM. Highest mean values are underlined. A star (⋆) marks datasets with original node-level features, for which K-Means is included as an additional baseline.

dataset, the performance of TGC matched that of plain Node2Vec — meaning the former was not able to further improve the node embeddings previously obtained by the latter. On the other hand, DAEGC resulted in an out-of-memory error due to its implementation relying on dense matrices to compute the reconstruction loss, therefore rendering it unfeasible for larger networks. We also note that, although tNodeEmbed achieved the highest accuracy on the same dataset, its performance was suboptimal compared to both Node2Vec and TGC when considering the other metrics.

Most interestingly, for the *Brain* and *Patent* datasets, the best results were instead obtained by simply employing spectral clustering on the graph's Laplacian. These results are worth highlighting because spectral techniques are among the methods recognized to be asymptotically optimal down to the detectability threshold of communities in graphs [18], considering solely their topology — which may be further improved by models exploiting a graph's attributes and temporal dynamics as well [7]. For the *arXivAI* and *DBLP* datasets, in contrast, the algorithm was one of the most underperforming solutions. This may be explained by the high number of connected components (disjoint subgraphs) in both networks, described in Table 2, which is expected to impact the graph's partitioning, as it depends on its matrix's eigendecomposition — alternatively, employing it on the largest connected component only may allow to better assess its performance in such cases.

While we have chosen to report the results obtained in our tests with DynNode2Vec, we note that its worse performance might be due to a lack of hyperparameter tuning, as we chose to employ the same set of hyperparameters used for Node2Vec and tNodeEmbed, to allow a direct comparison among the three models for network representation learning. It is also likely that the amount of snapshots with very few nodes available for walk sampling in some datasets has negatively impacted the expressiveness of the obtained node-level embeddings. In this case, its performance may be further improved by utilizing a different strategy to split the graph into snapshots — e.g., by sequentially merging subsequent snapshots or selecting a predefined time interval to construct them instead.

Meanwhile, the results from tNodeEmbed — which employs a similar, but more sophisticated approach to combine temporal node embeddings, by solving the closest orthogonal approximation problem for each sequential pair of graph snapshots — did not display the same limitation, except on the *School* dataset, in which both models performed poorly. Their worsening performance is possibly due to the continuous-time temporal dynamics of the graph resulting in a very large number of snapshots with few nodes and edges, as observed in Table 2. Given that plain Node2Vec's performance considering the same dataset as a static graph was the best among all models, it is likely that selecting an arbitrary time interval to construct the snapshots may potentially contribute to improve tNodeEmbed's efficiency and effectiveness, as discussed for DynNode2Vec. Overall, these results highlight how model selection also depends on the choice of representing a network as a static, snapshot-based, or event-based temporal graph, and that the choice of the best approach may be highly dependent on the dataset's characteristics — although, given the apparent benefits and potential pitfalls of considering the network's temporal evolution for SkipGram-based encoders, a one-size-fits-all strategy that is universally optimal for all datasets seems unlikely at first.

Dataset	Model	ACC	AMI	ARI
PubMed	K-Means	**.682 ± .001**	*.235 ± .002*	**.272 ± .001**
	DAEGC	*.690 ± .010*	***.257 ± .012***	*.285 ± .020*
	TGC	.678 ± .002	.221 ± .005	.254 ± .002

Table 4: Inductive evaluation. We mark the best results in the test set in bold and italic, and underline the highest mean values.

Lastly, our evaluation in an **inductive learning setting** was restricted to a single dataset due to the constraints previously discussed in Section 3.3. We employed a temporal split of approximately 60%/20%/20% nodes for the train, validation, and test sets, respectively, with each set generated by sequentially aggregating snapshots, so to guarantee that the train and validation sets would not include any information from future nodes. The validation set was used for hyperparameter tuning and early stopping only, and the test set to assess the model's performance afterwards.

We observe that both DAEGC and K-Means achieved better metrics than TGC in this setting. We initially consider this may be due to the temporal nature of the dataset: the dynamics of citation networks are overall very simple, which may hinder the extent to which a model can effectively exploit this information to improve its learned representations. Moreover, all models were less performant than in a transductive setting when evaluated on metrics adjusted by chance, although more tests would be necessary to compare their efficiency on different datasets and better understand how well they generalize to unseen nodes in different application scenarios.

5 Conclusion

Recent trends in graph research with machine learning show an increasing focus on dynamical graphs. In this paper, we compared the performance of various models to assess the effectiveness of current state-of-the-art methods for node clustering. Our results showed that, although generally understood in the literature as the current state of the art, GNN-based architectures introduced specifically for node-level clustering may still underperform when compared to more established approaches, especially on unattributed graphs. We argue there is still considerable margin for improvement in this area, such as further investigations on how to better exploit the temporal dynamics of a graph to learn more expressive node representations. Likewise, a better understanding of how GNNs compare to optimal approaches w.r.t. the detectability threshold of communities is highly desired, as well as further tests to evaluate their performance considering an inductive learning setting. In the near future, we aim to explore these topics and related questions in the context of node-level clustering of temporal graphs, and to propose solutions to improve their benchmarking using synthetic graphs with node-level features and known community structures.

Acknowledgments

We thank the members of the Inverse Complexity Lab and the Department of Network and Data Science at the Central European University, Vienna, for their guidance and support during the author's stay in their Doctoral Support Program, at which time this paper was written. We also thank the anonymous reviewers for their valuable input and welcomed suggestions during its review.

References

[1] Shaked Brody, Uri Alon, and Eran Yahav. 2022. How Attentive are Graph Attention Networks? arXiv:2105.14491 [cs.LG]

[2] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *2014 Conference on Empirical Methods in Natural Language Processing EMNLP*. Association for Computational Linguistics, 1724–1734. https://doi.org/10.3115/v1/D14-1179

[3] Nelson Aloysio Reis de Almeida Passos, Emanuele Carlini, and Salvatore Trani. 2024. PubMed-Temporal: a dynamic graph dataset with node-level features. https://doi.org/10.5281/zenodo.13932076

[4] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

[5] Santo Fortunato. 2010. Community Detection in Graphs. *Physics Reports* 486, 3-5 (2 2010), 75–174. https://doi.org/10.1016/j.physrep.2009.11.002

[6] F.A. Gers, J. Schmidhuber, and F. Cummins. 1999. Learning to forget: continual prediction with LSTM. In *Ninth International Conference on Artificial Neural Networks ICANN*, Vol. 2. 850–855. https://doi.org/10.1049/cp:19991218

[7] Amir Ghasemian, Pan Zhang, Aaron Clauset, Cristopher Moore, and Leto Peel. 2016. Detectability Thresholds and Optimal Algorithms for Community Structure in Dynamic Networks. *Physical Review X* 6, 3 (July 2016). https://doi.org/10.1103/physrevx.6.031005

[8] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural Message Passing for Quantum Chemistry. In *34th International Conference on Machine Learning - Volume 70* (Sydney, NSW, Australia) *(ICML'17)*. JMLR.org, 1263–1272.

[9] Daniele Grattarola, Daniele Zambon, Filippo Maria Bianchi, and Cesare Alippi. 2022. Understanding Pooling in Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems* (2022), 1–11. https://doi.org/10.1109/TNNLS.2022.3190922

[10] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *22nd ACM SIGKDD*. ACM. https://doi.org/10.1145/2939672.2939754

[11] Jin Guo, Zhen Han, Su Zhou, Jiliang Li, Volker Tresp, and Yuyi Wang. 2022. Continuous Temporal Graph Networks for Event-Based Graph Data. In *DLG4NLP 2022*. Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.dlg4nlp-1.3

[12] Ehsan Hajiramezanali, Arman Hasanzadeh, Nick Duffield, Krishna R Narayanan, Mingyuan Zhou, and Xiaoning Qian. 2020. Variational Graph Recurrent Neural Networks. arXiv:1908.09710 [cs.LG]

[13] Bronwyn Hall, Adam Jaffe, and Manuel Trajtenberg. 2001. *The NBER Patent Citation Data File: Lessons, Insights and Methodological Tools*. https://doi.org/10.3386/w8498

[14] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc.

[15] Alan G. Hawkes. 1971. Point Spectra of Some Mutually Exciting Point Processes. *Journal of the Royal Statistical Society Series B: Statistical Methodology* 33, 3 (Oct. 1971), 438–443. https://doi.org/10.1111/j.2517-6161.1971.tb01530.x

[16] Thomas N Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. *NIPS Workshop on Bayesian Deep Learning* (2016).

[17] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR*. arXiv:1609.02907 [cs.LG]

[18] Florent Krzakala, Cristopher Moore, Elchanan Mossel, Joe Neeman, Allan Sly, Lenka Zdeborová, and Pan Zhang. 2013. Spectral redemption in clustering sparse networks. *Proceedings of the National Academy of Sciences* 110, 52 (Nov. 2013), 20935–20940. https://doi.org/10.1073/pnas.1312486110

[19] Pan Li and Jure Leskovec. 2022. Graph Neural Networks: Foundations, Frontiers, and Applications. Springer Singapore, Singapore.

[20] Meng Liu, Ke Liang, Yue Liu, Siwei Wang, Sihang Zhou, and Xinwang Liu. 2023. arXiv4TGC: Large-Scale Datasets for Temporal Graph Clustering. arXiv:2306.04962 [cs.AI] https://arxiv.org/abs/2306.04962

[21] Meng Liu, Yue Liu, Ke Liang, Wenxuan Tu, Siwei Wang, Sihang Zhou, and Xinwang Liu. 2024. Deep Temporal Graph Clustering. In *The 12th International Conference on Learning Representations*.

[22] Antonio Longa, Veronica Lachi, Gabriele Santin, Monica Bianchini, Bruno Lepri, Pietro Lio, Franco Scarselli, and Andrea Passerini. 2023. Graph Neural Networks for Temporal Graphs: State of the Art, Open Challenges, and Opportunities. *Transactions on Machine Learning Research* (2023).

[23] Sedigheh Mahdavi, Shima Khoshraftar, and Aijun An. 2018. dynnode2vec: Scalable Dynamic Network Embedding. In *2018 IEEE International Conference on Big Data (Big Data)*. IEEE. https://doi.org/10.1109/bigdata.2018.8621910

[24] Rossana Mastrandrea, Julie Fournet, and Alain Barrat. 2015. Contact Patterns in a High School: A Comparison between Data Collected Using Wearable Sensors, Contact Diaries and Friendship Surveys. *PLOS ONE* 10, 9 (Sept. 2015), e0136497. https://doi.org/10.1371/journal.pone.0136497

[25] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781 [cs.CL]

[26] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. 2019. Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks. In *Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence* (Honolulu, Hawaii, USA) *(AAAI'19/IAAI'19/EAAI'19)*. AAAI Press, Article 565, 8 pages. https://doi.org/10.1609/aaai.v33i01.33014602

[27] Galileo Mark Namata, Ben London, Lise Getoor, Bert Huang, and U Edu. 2012. Query-driven active surveying for collective classification. In *10th International Workshop on Mining and Learning with Graphs*, Vol. 8. MLG.

[28] Mark Newman. 2018. *Networks* (2 ed.). Oxford University Press.

[29] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao B. Schardl, and Charles E. Leiserson. 2019. EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs. arXiv:1902.10191 [cs.LG]

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[31] Tiago P. Peixoto. 2019. Bayesian Stochastic Blockmodeling. , 289-332 pages. https://doi.org/10.1002/9781119483298.ch11

[32] Maria Giulia Preti, Thomas AW Bolton, and Dimitri Van De Ville. 2017. The dynamic functional connectome: State-of-the-art and perspectives. *NeuroImage* 160 (Oct. 2017), 41–54. https://doi.org/10.1016/j.neuroimage.2016.12.061

[33] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. 2020. Temporal Graph Networks for Deep Learning on Dynamic Graphs. arXiv:2006.10637 [cs.LG]

[34] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. 2020. DySAT. In *13th International Conference on Web Search and Data Mining*. ACM. https://doi.org/10.1145/3336191.3371845

[35] Uriel Singer, Ido Guy, and Kira Radinsky. 2019. Node embedding over temporal graphs. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence* (Macao, China) *(IJCAI'19)*. AAAI Press, 4605–4612.

[36] V. A. Traag, L. Waltman, and N. J. van Eck. 2019. From Louvain to Leiden: guaranteeing well-connected community. *Scientific Reports* 9, 1 (26 3 2019), 5233. https://doi.org/10.1038/s41598-019-41695-z

[37] Anton Tsitsulin, John Palowitch, Bryan Perozzi, and Emmanuel Müller. 2024. Graph Clustering with Graph Neural Networks. *J. Mach. Learn. Res.* 24, 1, Article 127 (mar 2024), 21 pages.

[38] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. arXiv:1710.10903 [stat.ML]

[39] Nguyen Xuan Vinh, Julien Epps, and James Bailey. 2009. Information theoretic measures for clusterings comparison: is a correction for chance necessary?. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML '09)*. ACM. https://doi.org/10.1145/1553374.1553511

[40] Chun Wang, Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Attributed Graph Clustering: A Deep Attentional Embedding Approach. In *Twenty-Eighth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization. https://doi.org/10.24963/ijcai.2019/509

[41] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems* 32, 1 (2021), 4–24. https://doi.org/10.1109/TNNLS.2020.2978386

[42] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. 2020. Inductive Representation Learning on Temporal Graphs. arXiv:2002.07962 [cs.LG]

[43] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks? arXiv:1810.00826 [cs.LG]

[44] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48* (New York, NY, USA) *(ICML'16)*. JMLR.org, 40–48.

[45] Yuan Zuo, Guannan Liu, Hao Lin, Jia Guo, Xiaoqian Hu, and Junjie Wu. 2018. Embedding Temporal Network via Neighborhood Formation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '18)*. ACM. https://doi.org/10.1145/3219819.3220054