Proposal of Dynamic Community Detection in Time Series via Transformer-based Graph Neural Networks

Tianyang Luo
tluo11@illinois.edu

03/18/2025

1 Introduction

Community detection is the process of identifying groups or clusters of nodes within a network, wherein nodes within the same group have denser connections with each other than with nodes outside the group. This method is instrumental in uncovering hidden structural information across various complex networks, including social networks, biological systems, and financial markets (Fortunato, 2010) [1].

Traditionally, community detection methodologies primarily include modularity maximization techniques, hierarchical clustering, and statistical inference methods. For instance, Newman's spectral method, a representative modularity-based algorithm, identifies communities by optimizing the modularity measure, which evaluates the density of links within clusters compared to links between clusters (Newman, 2004) [2]. Nevertheless, these traditional techniques confront significant technical bottlenecks. Notably, modularity maximization encounters resolution limits, making it difficult to detect smaller communities within larger networks. Moreover, scalability issues become evident as these algorithms often fail to efficiently handle extensive real-world networks. Additionally, traditional methods are sensitive to network noise and hyperparameter settings, resulting in inconsistent and unstable community structures under slight perturbations or changes in parameters (Fortunato, 2010) [1].

Recent advances in deep learning have facilitated Graph Neural Networks (GNNs) as a promising alternative to traditional community detection methods. GNN-based models operate by learning adaptive embeddings that integrate both local graph structure and rich node features into node representations in an end-to-end manner (Kipf & Welling, 2017) [3]. Compared to classical approaches, GNN methods have several notable advantages: First, they provide nuanced node embeddings by utilizing detailed node and edge features, facilitating richer representation of network structures. Second, GNN architectures, through their inherent adaptability, effectively capture nonlinear relationships within networks, enhancing the separation and detection accuracy of communities (Li et al., 2020) [4]. Lastly, due to their compatibility with modern deep-learning frameworks, GNNs exhibit better scalability for large-scale networks compared to conventional algorithms.

In financial markets and transportation systems, the importance of effective community detection is particularly pronounced. Financial markets, such as stock networks, exhibit intricate interactions among assets. Community detection can reveal clusters of stocks displaying similar behaviors or dynamics, thereby informing portfolio diversification strategies, risk management, and broader insights into market dynamics (Mantegna, 1999) [5]. In transportation networks, community detection methodologies identify functional regions or hubs, assisting planners in pinpointing traffic bottlenecks, optimizing urban mobility, and improving network efficiency (Huang et al., 2018) [6].

Although dynamic community detection, which examines communities within evolving or time-dependent networks, has demonstrated practical relevance by tracking structural shifts and capturing emerging or dissolving communities, there remains a small proportion of research on this topic, and even fewer studies adequately address the multivariate time-series scenarios common in financial markets (Rossetti & Cazabet, 2018) [7].

To bridge this critical research gap, this study proposes a transformer-based graph convolution module, learnable enhanced layers combined with convolutional autoencoders to dynamically analyze networks constructed from multivariate time-series data. This approach fundamentally builds upon the representation learning paradigm, leveraging latent data representations to uncover complex interdependencies across multiple time series.

Theoretically, this combined spatial-temporal model could achieve better performance in dynamic community detection due to several advantages. First, the transformer-based graph convolution module effectively captures evolving temporal relationships by leveraging self-attention mechanisms, allowing the model to adaptively track changing connections over time. Second, learnable enhanced layers provide flexibility, enabling the model to emphasize crucial features dynamically and filter out irrelevant noise. Finally, convolutional autoencoders compress high-dimensional data into meaningful lower-dimensional representations, reducing complexity and highlighting essential community structures.

These integrated strengths help the proposed model more accurately identify and understand dynamic communities within time-series networks, thereby enhancing the precision of decision-making processes related to portfolio management, risk assessment, and financial forecasting.

2 Literature Review

In recent years, the application of Graph Neural Networks (GNNs) has significantly advanced the field of community detection, particularly within dynamic and complex networks such as financial systems. These models aim to address limitations found in traditional methods by effectively capturing both structural and temporal nuances inherent in evolving networks.

One of the early approaches is the Graph Neural Network Inspired Algorithm for Unsupervised Network Community Detection (Sobolevsky & Belyi, 2022) [8]. This method integrates GNNs with modularity optimization to detect communities without relying on external labels. While it demonstrates applicability to temporal data, such as daily taxi ridership networks, the model does not explicitly incorporate temporal dependencies, potentially limiting its responsiveness to rapid changes in community structures.

Around the same period, Qiu et al. (2022) proposed VGAER [9], a variational graph autoencoder reconstruction method specifically designed for community detection. Although VGAER integrates network structure and node features to extract modularity information and achieves notable improvements in clustering performance without requiring external labels, it does not account for time-series data.

Subsequently, the Dynamic Community Detection Method for Complex Networks Based on Deep Self-Coding Network [10] was introduced by Zhang and Xiao (2022). This approach employs an encoding-decoding mechanism to reconstruct node features, aiming to mitigate the oversmoothing problem common in GNNs. However, its validation on relatively small datasets raises questions about its scalability to larger, real-world dynamic graphs.

Considering these gaps, it becomes meaningful and necessary to design a new algorithm architecture specifically tailored for dynamic community detection from multivariate time-series data.

Moreover, several recent studies have successfully transitioned deep graph models from static to dynamic settings. For instance, Pareja et al. (2020) introduced EvolveGCN [11], a model in which the network's weights evolve over time through a recurrent mechanism, instead of sharing a fixed set of weights across all time windows. This dynamic evolution of weights may prevent issues commonly observed in static models, where fixed weight matrices can lead to suboptimal community detection results within specific temporal intervals.

Additionally, other studies have introduced learnable components into adjacency matrices. For example, Graph WaveNet (Wu et al., 2019) [12] proposed an adaptive dependency matrix learned through node embeddings. By dynamically adjusting the underlying adjacency structure and applying dilated causal convolutions for long-range temporal modeling, Graph WaveNet captures hidden spatial dependencies beyond static graph connections, potentially enhancing the effectiveness of community detection tasks.

3 Stage1 Progress (Until 03/16/2025)

This study has achieved a static transformer-based graph convolution module, learnable enhanced layers combined with convolutional autoencoders to dynamically analyze networks constructed from multivariate time-series data.

3.1 Algorithm Architecture

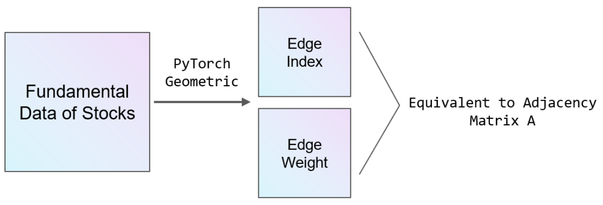

Figure 1: Graph Creation

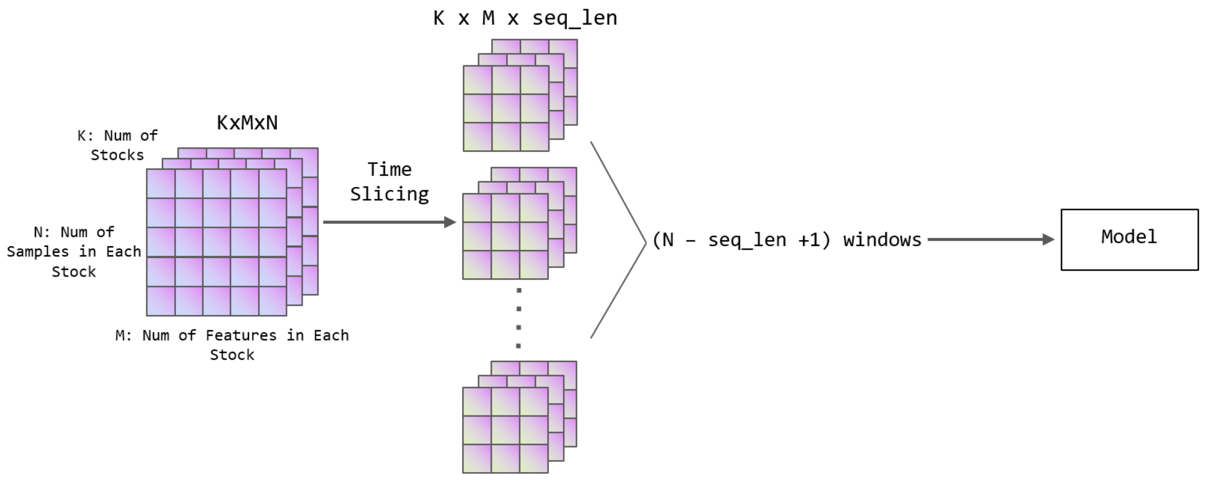

Figure 2: Time Slicing

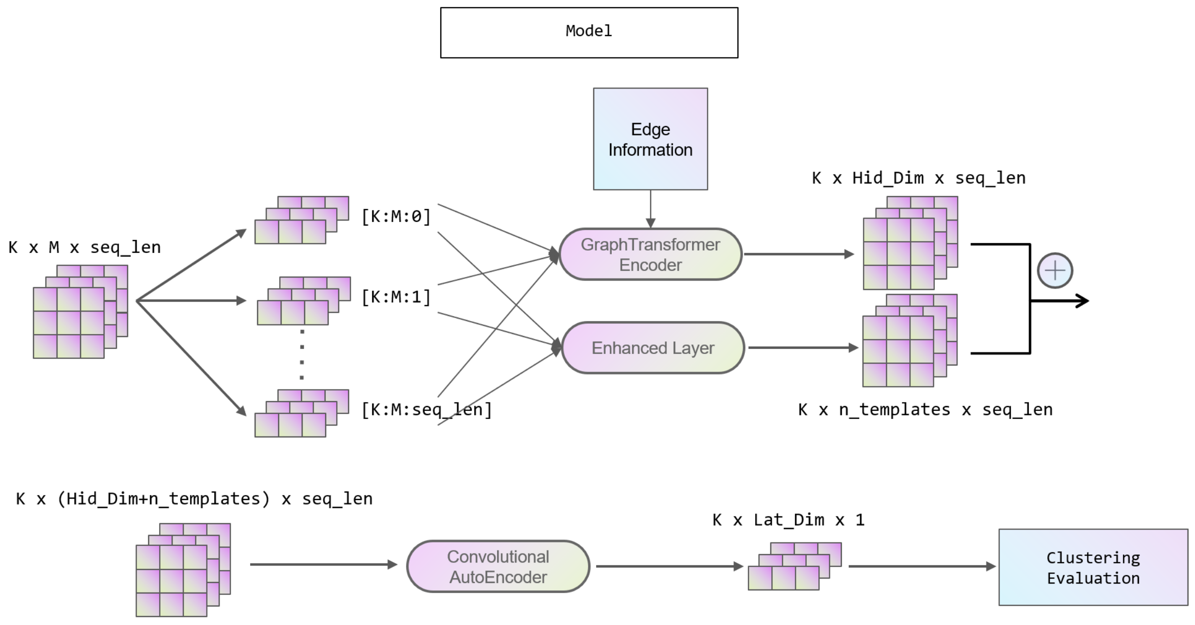

Figure 3: Model & Evaluation

3.2 Results

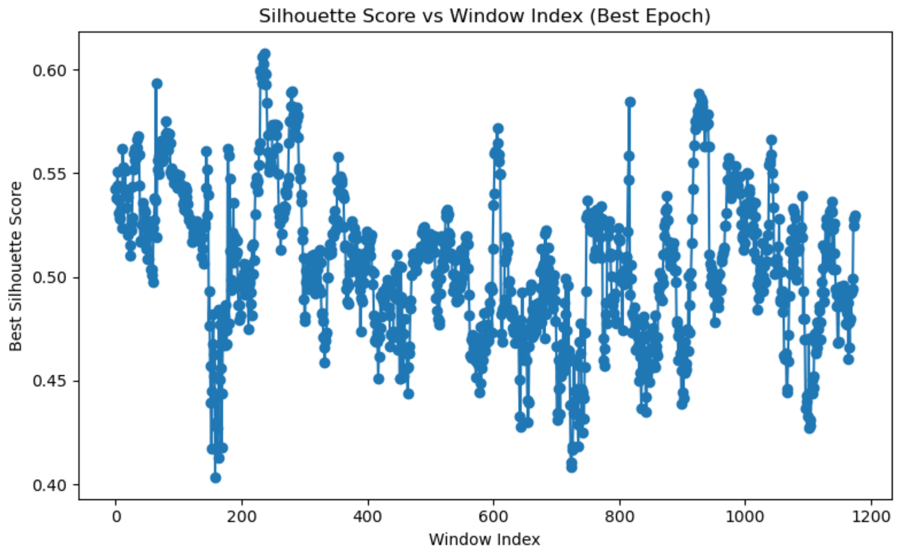

Figure 4: Time-Varying Silhouette Score in the Best Epoch

It is encouraging that the proposed model achieves an average silhouette coefficient that is relatively high (above 0.5) in many time windows. However, some intervals exhibit noticeably lower performance, indicating areas where the model may still have room for improvement. These lower-performance intervals suggest that the use of fixed model weights and static adjacency matrices across all time windows might limit the model's ability to adapt effectively to changing conditions. To address this issue, this study identifies the need to explore methods that specifically enhance performance during these intervals, ultimately aiming to improve the robustness and overall accuracy of dynamic community detection.

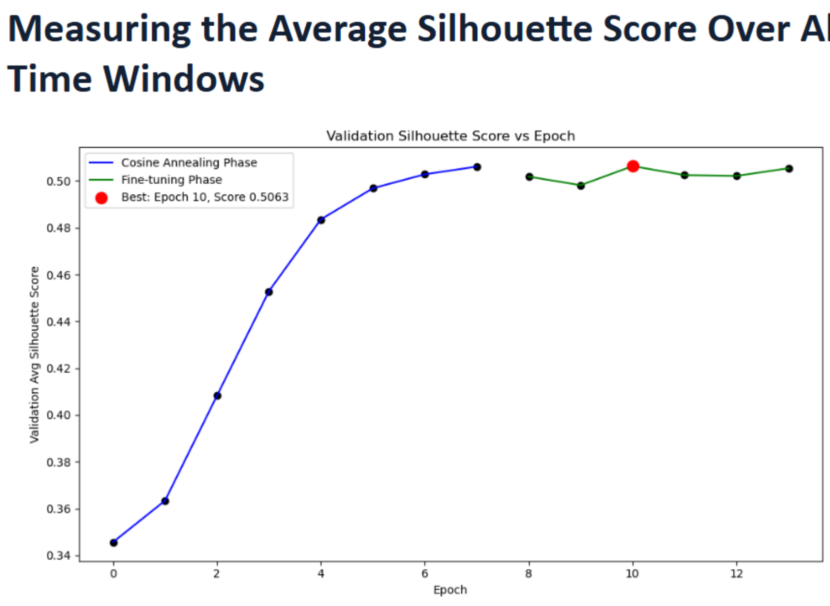

Figure 5: Training and Validation Process

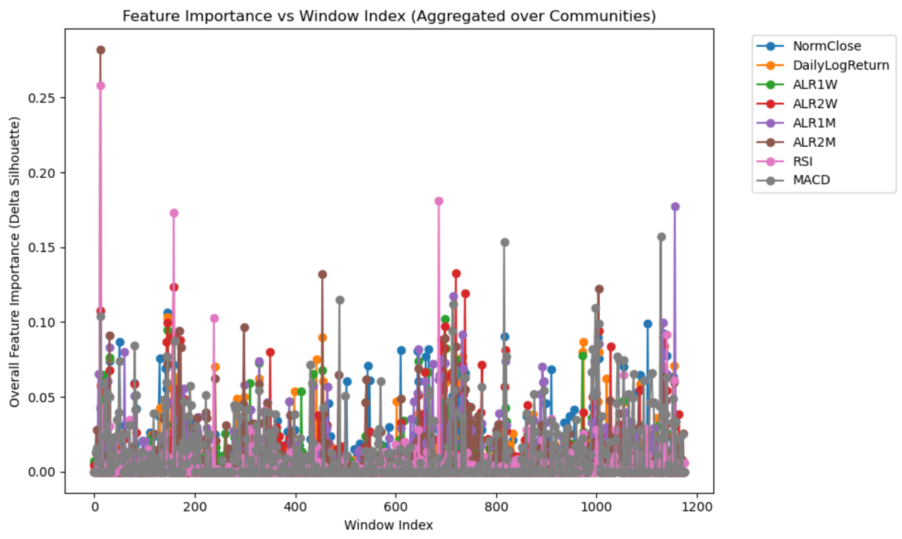

Figure 6: Time-Varying Feature Importance in the Best Epoch

Time-Varying Best N_Clusters

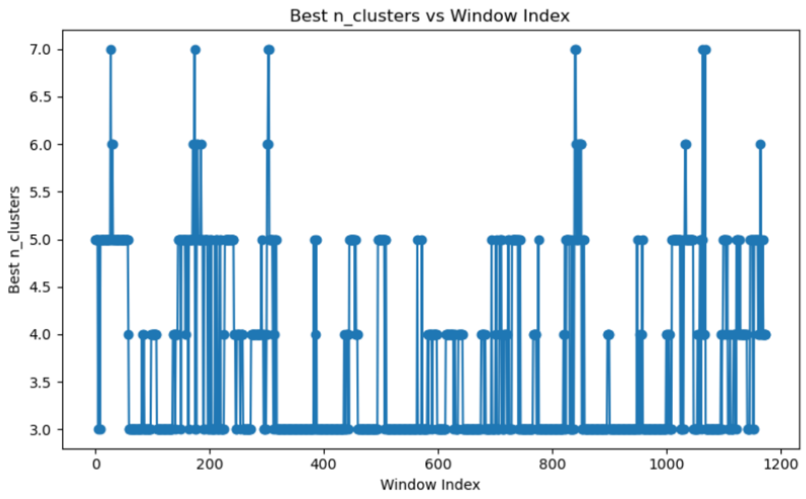

Figure 7: Best n clusters of All Time Steps in Each Epoch

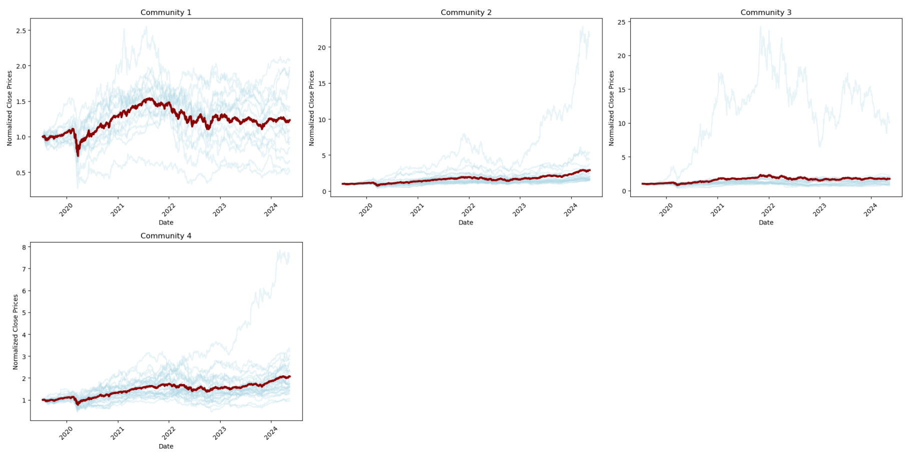

Figure 8: Clustering Visualization of the Last Window

4 Stage2 Progress (Until 04/02/2025)

After improving the convolutional autoencoder module by adding channel attention and temporal attention, and implementing an original learnable matrix using low-rank parameter matrices and a sparsity threshold to generate a sparse, adaptive adjacency representation, the average silhouette score increased to 0.5640 (an improvement of over 10%), and its standard deviation dropped to 0.0235 (a decrease of about 50%); additionally, the optimal number of clusters is no longer consistently 3 at most time points, significantly mitigating the bias of the silhouette score toward a small number of clusters, achieving more refined community detection, and providing more precise and detailed opportunities for subsequent pairs trading and hedging.

4.1 Results

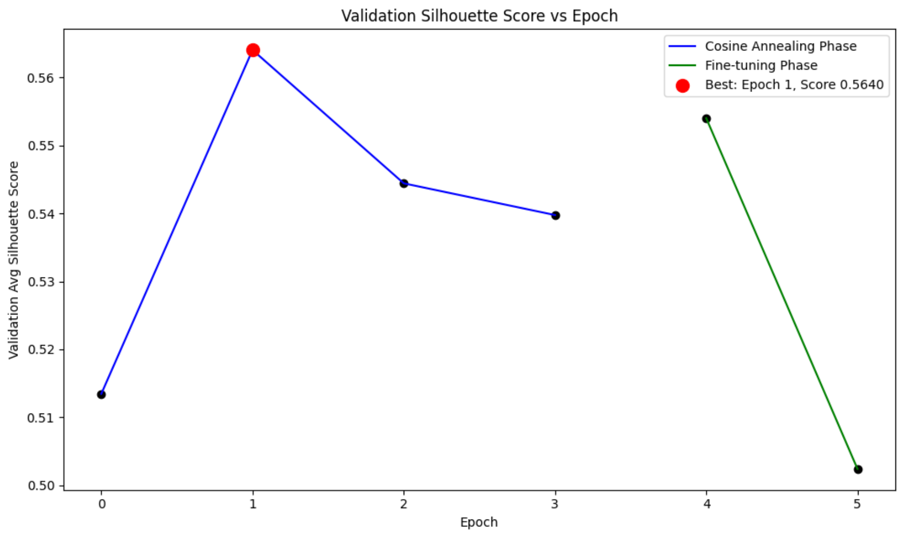

Figure 9: Training and Validation Process

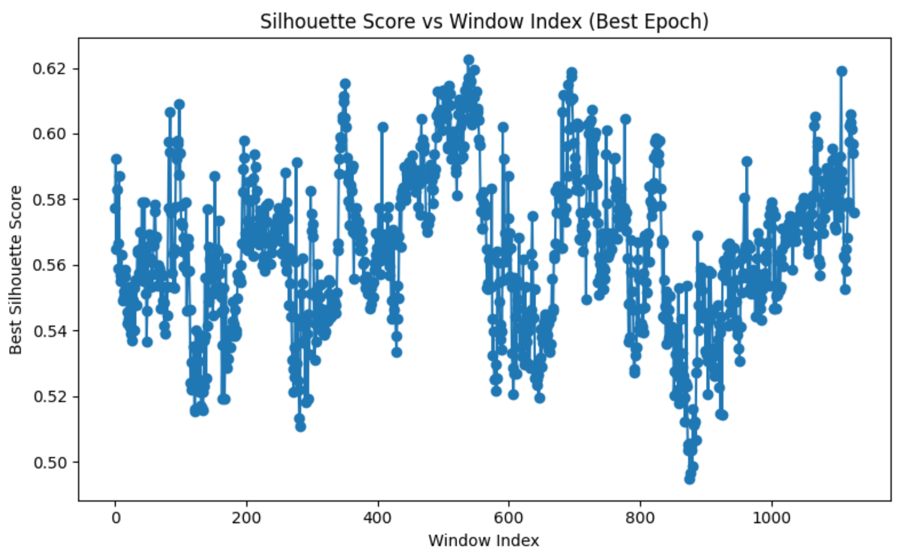

Figure 10: Average Silhouette Score of All Time Steps in Each Epoch

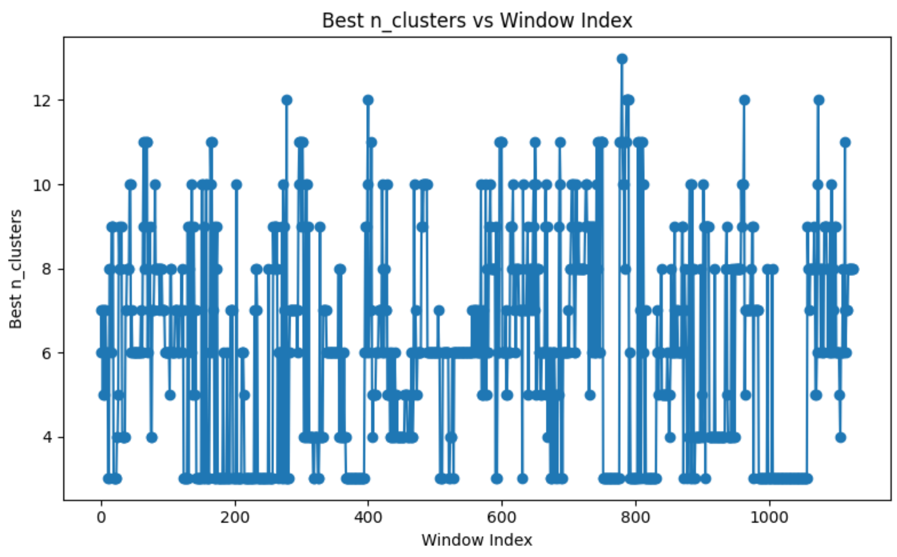

Figure 11: Best n clusters of All Time Steps in Each Epoch

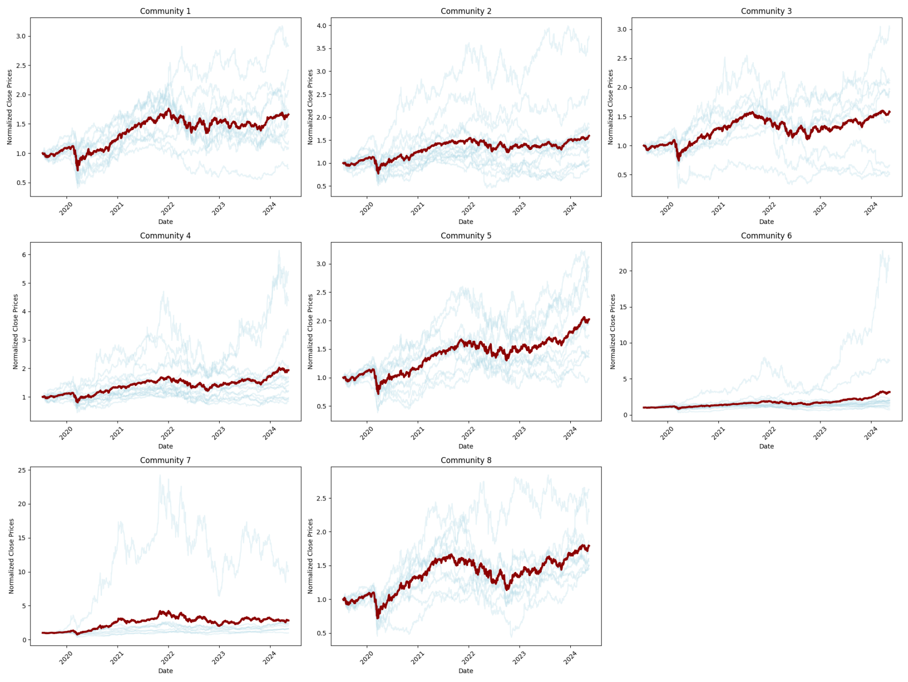
Figure 12: Clustering Visualization of the Last Window

5 Research Questions

5.1 Dynamic Weight Evolution via Fusion

Challenge: The Transformer-based convolution module's static self-attention weights lack temporal adaptability for dynamic graph evolution. Simply applying EvolveGCN's RNN-based parameter evolution to these attention weights poses integration risks, as it requires a careful mechanism to ensure consistency across multiple attention heads and to maintain numerical stability.

5.1.1 Technical Approach:

- *Per-Head Hidden States:* For each multi-head attention layer l with H heads, maintain separate hidden states for query and key matrices. Denote them as $\big(W_{Q,t}^{(l,h)},\, W_{K,t}^{(l,h)}\big)$ for head $h \in \{1, \ldots, H\}$ at time t. These hidden states are updated via an RNN cell (GRU or LSTM) that processes both the previous states and the current node embeddings:

$$\big(W_{Q,t}^{(l,h)}, W_{K,t}^{(l,h)}\big) = \mathrm{RNN}\Big(\big(W_{Q,t-1}^{(l,h)},\, W_{K,t-1}^{(l,h)}\big),\, H_t^{(l-1)}\Big), \tag{1}$$

  where $H_t^{(l-1)}$ is the node embedding output from the preceding layer $(l-1)$ at time t. This design ensures each attention head evolves independently while still leveraging the shared embedding context.

- *Dimension Alignment and Initialization:* Flatten or reshape the query/key matrices into 1D vectors before feeding them into the RNN cell. After the RNN updates, reshape them back to their multi-head attention dimensions. For stability, initialize each RNN's hidden states with the static solution from a pre-trained Transformer-based module, ensuring that the model starts from a known baseline.

- *Sequential Update Pipeline:* At each time step t, the pipeline proceeds as follows:
  1. Obtain node embeddings $H_t^{(l-1)}$ from the previous layer or time step.
  2. For each attention head h, update $(W_{Q,t}^{(l,h)}, W_{K,t}^{(l,h)})$ via the RNN cell.
  3. Compute the attention logits:

$$\alpha_t^{(l,h)} = \frac{\left(H_t^{(l-1)} W_{Q,t}^{(l,h)}\right)\left(H_t^{(l-1)} W_{K,t}^{(l,h)}\right)^\top}{\sqrt{d}}. \tag{2}$$

  4. Aggregate over heads, apply softmax, and compute the updated node embeddings $H_t^{(l)}$.

5.1.2 Implementation Notes for Compatibility:

- *Learning Rate Scheduling:* Employ a smaller learning rate for RNN parameters than for the Transformer-based module to reduce gradient conflict and ensure smoother updates.
- *Gradient Clipping:* Clip gradients within the RNN to handle large updates that may destabilize the attention mechanism.
- *Multi-Stage Training:* Optionally freeze the RNN parameters in early epochs to let the Transformer-based module converge, then unfreeze and co-train them to gradually incorporate dynamic evolution.

5.1.3 Theoretical Support:

- The Lipschitz continuity of the RNN update (as shown in EvolveGCN's Theorem 1) helps control gradient explosion, supporting stable weight evolution.
- Per-head hidden states respect multi-head independence while allowing each head to capture distinct temporal patterns in the graph's evolution.

5.2 Dynamic Relationship Modeling via Fusion (alternative option)

5.2.1 Challenge:

The Transformer-based convolution module's attention scores implicitly model relationships but lack explicit spatial dependency learning. GraphWaveNet's adaptive adjacency requires integration with the attention mechanism without disrupting global context capture.

5.2.2 Technical Approach:

Inject learnable node embeddings $\mathbf{E}_1, \mathbf{E}_2$ into the attention score computation:

$$\tilde{A}_{adp} = \text{Softmax}\left(\text{ReLU}(\mathbf{E}_1\mathbf{E}_2^\top)\right) \tag{3}$$

Fuse with original attention scores α_{ij}:

$$\alpha_{ij}^{\text{fused}} = \lambda\alpha_{ij} + (1-\lambda)\tilde{A}_{adp}[i,j] \tag{4}$$

where λ is a learnable mixing coefficient.

5.2.3 Theoretical Support:

The hybrid adjacency matrix satisfies spectral graph convolution properties (Theorem 2 in GraphWaveNet) while preserving Transformer's expressiveness through multi-head attention.

5.3 Expected Formulations

- **Evolved Transformer-based convolution Layer:**

$$H_t^{(l+1)} = \text{Softmax}\left(\frac{(H_t^{(l)} W_{Q,t}^{(l)})(H_t^{(l)} W_{K,t}^{(l)})^\top}{\sqrt{d}}\right) H_t^{(l)} W_{V,t}^{(l)} \tag{5}$$

  with $W_{\{Q,K\},t}^{(l)}$ updated via Equations 1 or 2.

- **Adaptive-Enhanced Attention:**

$$Z = \sum_{h=1}^{H} \left(\alpha_h^{\text{fused}} V_h\right) + \text{MLP}\left(\sum_{k=0}^{K} \tilde{A}_{adp}^{k} X W_k\right) \tag{6}$$

  combining Transformer's multi-head outputs with graph diffusion from adaptive adjacency.

5.4 Key Challenges

- *Training Stability:* Co-training RNNs with Transformers risks gradient conflicts. Solution: Layer-wise curriculum learning.
- *Complexity Control:* Adaptive adjacency increases $O(N^2)$ costs. Mitigation: Use low-rank embeddings $\mathbf{E}_1, \mathbf{E}_2 \in \mathbb{R}^{N \times c}$ where $c \ll N$.
- *Temporal-Spatial Alignment:* Ensure dilated TCNs (from GraphWaveNet) align with Transformer's positional encoding. Proposed fix: Replace TCNs with temporal attention heads.

5.5 Improve Enhanced Layer (alternative option)

This study is actively exploring alternative metrics or computational strategies (e.g., cosine similarity or Mahalanobis distance) to better capture the nuances of global dependencies without significantly increasing complexity.

5.6 Evaluation Metrics

5.6.1 Justification for Silhouette Coefficient in Financial Data

When evaluating clustering results on financial data, choosing appropriate metrics that reflect natural groupings and perform consistently across various market conditions is essential. The silhouette coefficient, which measures clustering quality by assessing both intra-cluster cohesion and inter-cluster separation, has been identified as a particularly effective metric. Several significant studies provide strong evidence supporting its applicability in financial contexts.

Turner (2021), in the paper "Graph Auto-Encoders for Financial Clustering" [13], demonstrated that the silhouette coefficient effectively evaluates the quality of clusters derived from financial time series data. Turner highlighted that clusters exhibiting higher silhouette scores correlated strongly with known market segments, thus validating the metric's suitability as an unsupervised evaluation tool for financial clustering tasks.

Similarly, Wirth et al. (2024), in their research titled "Longitudinal Market Structure Detection Using a Dynamic Modularity-Spectral Algorithm" [14], applied the silhouette coefficient to assess dynamic clustering performance in stock market data. Their findings revealed that although modularity is commonly employed to evaluate network partitions, it may not adequately capture the nuanced distinctions within financial datasets. In contrast, the silhouette coefficient provided clearer insights into cluster stability over time, thereby facilitating more effective portfolio diversification strategies.

Additionally, Cai et al. (2016), in their comprehensive review "Clustering Approaches for Financial Data Analysis: A Survey" [15], examined various clustering evaluation metrics specifically in financial contexts. They underscored the silhouette coefficient's effectiveness in detecting natural groupings within noisy financial datasets, especially compared to modularity-based metrics, which heavily depend on network structures and may struggle when data characteristics are not strongly network-oriented.

Collectively, these studies demonstrate that the silhouette coefficient provides a robust theoretical basis and practical effectiveness for assessing clustering quality in financial data. Although modularity is valuable within network science, its application in financial data can be unreliable due to its assumptions of static network structure and sensitivity to market noise and rapid temporal changes—features inherently present in financial markets. Consequently, the silhouette coefficient emerges as a more reliable and versatile measure for evaluating clustering performance in financial applications.

5.6.2 Improvement on Current Evaluation Metric

One potential improvement to the current evaluation metric proposed by this study is to partition the original dataset into multiple segments, referred to as time-series k-folds, and then independently train and evaluate the model on each fold. Specifically, this study suggests calculating the average silhouette coefficient for all time windows within each fold and subsequently computing a weighted average across all folds. Furthermore, inspired by the principles of Average Performance (AP) and Average Forgetting (AF) [16] introduced by Zhang, Song, and Tao (2022), this study proposes measuring changes in the model's performance on previously learned tasks as new data is introduced, as well as quantifying the degree of forgetting of previously acquired knowledge. These metrics can then be integrated into the final evaluation measure for each epoch.

This proposed evaluation approach better captures the realistic evolution of data, offering a comprehensive view of how the model's performance evolves as new data becomes available. Compared with the current evaluation practice, which computes the mean silhouette coefficient at the last time window, the proposed method provides more detailed insights into the model's performance dynamics over time. Nonetheless, this study acknowledges potential challenges, including carefully designing an appropriate evaluation strategy and addressing increased computational complexity and longer training durations.

6 Expected Time Line

Timeline	**Progress**
Before April 15, 2025	Attempt to migrate the EvolveGCN idea to achieve a dynamic weight matrix.
Before May 7, 2025	Attempt to migrate the Graph WaveNet idea to achieve a learnable adjacency matrix.
Before May 31, 2025	Experiment with the alternative evaluation metric.
Before June 7, 2025	Determine the optimal combination of advanced techniques and establish a sound theoretical foundation.
After June 7, 2025	Conduct further research and complete the full paper writing.

Table 1: Research Time Table

7 Expected Outcomes

This study anticipates that the proposed transformer-based graph convolution module combined with convolutional autoencoders will achieve better performance compared to existing graph neural network (GNN)-based models for unsupervised community detection in time series data. Specifically, this study aims to demonstrate at least a 15% improvement in average silhouette scores compared to baseline GNN approaches. Additionally, the proposed model is expected to reduce the standard deviation of silhouette coefficients across different time windows by approximately 10%, indicating more stable community detection over time. Furthermore, by integrating dynamic components—such as time-varying adjacency matrices and adaptive model weights—this study expects an additional improvement of at least 5% in the average silhouette score relative to the static version. These results would confirm the effectiveness and potential of dynamic spatial-temporal modeling in capturing evolving relationships.

8 References

References

[1] Fortunato, S. 2010. Community detection in graphs. *Physics Reports*, 486(3–5), 75–174.

[2] Newman, M. E. J. 2004. Detecting community structure in networks. *The European Physical Journal B*, 38(2), 321–330.

[3] Kipf, T. N., & Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR).*

[4] Li, X., et al. 2020. Graph Neural Networks: A Review of Methods and Applications. *IEEE Transactions on Neural Networks and Learning Systems.*

[5] Mantegna, R. N. 1999. Hierarchical structure in financial markets. *The European Physical Journal B*, 11(1), 193–197.

[6] Huang, Y., Zhang, Z., Chen, G., & Liu, Y. 2018. Traffic Flow Prediction with Spatial-Temporal Graph Diffusion Network. *IEEE Transactions on Intelligent Transportation Systems.*

[7] Rossetti, G., & Cazabet, R. 2018. Community Discovery in Dynamic Networks: A Survey. *ACM Computing Surveys*, 51(2), Article 35.

[8] Sobolevsky, S., & Belyi, A. 2022. Graph Neural Network Inspired Algorithm for Unsupervised Network Community Detection. *Applied Network Science*, 7(1), 1–18.

[9] Qiu, C., Huang, Z., Xu, W., & Li, H. 2022. VGAER: Graph Neural Network Reconstruction based Community Detection. *arXiv preprint arXiv:2201.04066.*

[10] Zhang, Z., & Xiao, X. 2022. A Dynamic Community Detection Method for Complex Networks Based on Deep Self-Coding Network. Available at `https://pubmed.ncbi.nlm.nih.gov/35958745/`.

[11] Pareja, A., Domeniconi, G., Chen, J., Ma, T., Suzumura, T., Kanezashi, H., Kaler, T., Schardl, T. B., & Leiserson, C. E. 2020. EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs. *arXiv preprint arXiv:1902.10191.*

[12] Wu, Z., Pan, S., Long, G., Jiang, J., & Zhang, C. 2019. Graph WaveNet for Deep Spatial-Temporal Graph Modeling. *arXiv preprint arXiv:1906.00121.*

[13] Turner, E. 2021. Graph Auto-Encoders for Financial Clustering. *arXiv preprint:2111.13519 .*

[14] Wirth, P., Medda, F., & Schröder, T. 2024. Longitudinal Market Structure Detection Using a Dynamic Modularity-Spectral Algorithm. *Under Review.*

[15] Cai, Y., Li, J., Hu, Y., & Zhang, J. 2016. Clustering Approaches for Financial Data Analysis: A Survey. *Journal of Finance and Data Science*, 2(1), 18–38.

[16] Zhang, X., Song, D., & Tao, D. 2022. CGLB: Benchmark Tasks for Continual Graph Learning. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022) Datasets and Benchmarks Track*.