VOL. 23 (1) (2026), 80-88 p-ISSN: 693-7554, e-ISSN:2654-3990

An Empirical Study of Temporal Graph Neural Networks for Dynamic Node Forecasting

Ricky Maulana Fajri[1], **Tasmi**[2], **Ni Wayan Priscillia Yuni Praditya**[3]

[1,2]Department of Computer System, Faculty of Computer and Natural Science, University of Indo Global Mandiri, South Sumatera, 30129, Indonesia

[3] Department of Information System, Faculty of Computer and Natural Science, University of Indo Global Mandiri, South Sumatera, 30129, Indonesia

Abstract

Abstract temporal graph modeling has become increasingly important for understanding and forecasting the dynamics of complex systems that evolve over time. One of the central challenges in temporal graph learning lies in identifying graph neural network (GNN) architectures that can effectively capture both spatial dependencies and temporal dynamics. This study presents a comprehensive benchmarking analysis of widely used GNN architectures, namely Graph Convolution Network (GCN), GraphSAGE, Graph Attention Network (GAT), Chebyshev Networks (ChebNet), and Simplified Graph Convolution Network (SGC), each integrated with recurrent mechanisms for temporal modeling. The evaluation is conducted on the WikiMaths dataset, a large-scale temporal graph dataset representing user visits of mathematics-related Wikipedia articles. Experimental results demonstrate that the choice of graph convolution operator significantly impacts temporal forecasting performance, with GraphSAGE and ChebNet consistently exhibiting superior performance compared to other architectures. This work provides empirical insights into the strengths and limitations of established temporal GNN models, contributing to a clearer understanding of their applicability in dynamic graph forecasting tasks.

Keywords: *Graph Neural Network; Temporal Graph; WikiMath; Time-series Prediction; Benchmarking*

1. Introduction

The increasing availability of data generated by interconnected systems has led to growing interest in graph-based learning approaches. Many real-world phenomena, such as information diffusion[1], [2], user activity patterns[3]-[5], and relational dynamics in online platforms[6], [7], can be naturally represented as graphs which node attributes evolve over time. Traditional time-series models, while effective for independent sequences, are limited in their ability to incorporate relational dependencies among entities such as graph characteristics[8].

Graph Neural Networks (GNNs) [9], [10]have emerged as a powerful framework for learning representations from graph-structured data. By leveraging neighbourhood aggregation, GNNs can capture complex relational patterns that are not accessible to conventional machine learning models[11]. However, most early GNN formulations focus on static graphs and are not designed to handle temporal dynamics inherent in many real-world applications[2], [9].

To address this limitation, temporal extensions of GNNs [12], [13] have been proposed by integrating recurrent or temporal convolutional mechanisms. These models aim to jointly capture spatial dependencies encoded in the graph structure and temporal dependencies arising from evolving node features. Despite their growing adoption, there remains limited empirical understanding of how different graph convolution operators perform under identical temporal modelling settings.

Existing studies [14] - [16] often introduce novel architectures or evaluate a limited subset of models, making it difficult to draw general conclusions about the relative strengths of established GNN variants. Specifically, Han et al[14] proposed STGCN a spatio-temporal convolutional model that captures spatial dependencies among locations and temporal user behaviour. While, Pareja et al [15] introduced introduced EvolveGCN that achieve high performance on temporal link prediction by implementing recurrent GCN architecture which they named dynamic graph. Finally, Xu et al [16] proposed a framework that learns time-

*Corresponding author. *E-mail address:*rickymaulanafajri@uigm.ac.id
Received: 23 December 2025, Accepted: 30 January 2026 and available online 31 January 2026
DOI: https://doi.org/10.33751/komputasi.v19i2.5260

ware node embedding that show a more robust spatio-temporal generalization. Moreover, Kazemi et al perform a comparisons which frequently conducted under heterogeneous experimental settings, that obscures the true impact of architectural choices on performance[17]. Despite these advances, prior studies focus on specific tasks or model designs and do not provide a systematic empirical comparison of temporal graph neural networks for dynamic node forecasting.

Motivated by these gaps, this study conducts a systematic empirical study of widely used GNN architectures within a unified temporal forecasting framework. Using the WikiMaths[18] dataset as a representative large-scale temporal graph, this work aims to provide empirical insights into how different neighbourhood aggregation strategies affect node-level temporal prediction accuracy. The remainder of this paper is organized as follows: Section 2 describes the dataset, experimental setup, evaluation protocol and presents the background of the study; Section 3 presents and discusses the empirical results; and Section 4 concludes the paper with limitations and directions for future research

2. Methods

2.1 Datasets

This study use Wikimath dataset [18], the dataset consists of a collection of important mathematics-related articles extracted from Wikipedia. It was publicly released during the development of the PyTorch Geometric Temporal framework[18] to support research in spatiotemporal graph learning. In this dataset, each node corresponds to a Wikipedia page, while edges represent hyperlink connections between pages. The resulting graph structure is static, directed, and weighted, capturing the inherent relational structure among mathematical topics.

Edge weights indicate the frequency of hyperlinks from a source page to a target page, reflecting the strength of their connection. The prediction target is the number of daily user visits to each Wikipedia page, recorded over a two-year period from March 16, 2019, to March 15, 2021. This temporal span yields 731 consecutive time steps, enabling the analysis of long-term temporal dynamics in user attention across the graph. The WikiMaths dataset poses a node-level regression task, where the objective is to predict future node values based on historical observations and graph connectivity. This formulation enables the analysis of how temporal dependencies and graph topology jointly influence predictive performance.

2.2 Datasets Cleaning and Preprocessing

Prior to model training, the dataset is processed using the standard pipeline provided by the *PyTorch Geometric Temporal* framework. Temporal sequences are constructed using a fixed lag window, where each input snapshot contains observations from the previous 14 time steps.

All node features are normalized to ensure numerical stability and faster convergence during training. No nodes or edges are removed, ensuring that the original graph topology is retained throughout the experiments.The dataset is then divided into training and test sets using a temporal split, where the first 50% of snapshots are used for training and the remaining 50% for testing. This split strategy prevents information leakage and reflects a realistic forecasting scenario.

2.3 Graph Construction

The WikiMaths dataset is modelled as a temporal attributed graph $G_t = (V, E, X_t)$ where V represents the set of nodes, E denotes the set of edges, and X_t corresponds to node features at time step t. The edge structure encodes hyperlinks between Wikipedia articles and remains fixed across time.

Each node is associated with a feature vector capturing historical activity patterns over the defined lag window. Edge weights, when available, are used to represent the strength of connections between nodes and are incorporated into graph convolution operations.

This formulation enables the models to jointly exploit spatial dependencies (graph structure) and temporal dependencies (node feature evolution). The resulting graph representation serves as the input to the proposed temporal GNN architectures.

2.4 Baselines

To perform an empirical study of temporal graph neural networks, several widely adopted graph convolutional architectures are evaluated within a unified temporal forecasting framework. The selected models include GCN[9], GraphSAGE[8], GAT[19], Chebyshev Network (ChebNet)[20], and Simplified Graph Convolution (SGC)[21], each coupled with a recurrent graph convolutional unit to enable temporal dependency modelling. These architectures represent diverse neighbourhood aggregation strategies commonly used in graph learning literature.

All evaluated models are implemented using a controlled and consistent experimental protocol, sharing identical hidden dimensions, learning rates, training epochs, and data splits. This standardized setup ensures a fair and unbiased comparison, allowing performance differences to be attributed solely to the architectural properties of the spatial convolution operators rather than to variations in optimization or hyperparameter tuning.

By benchmarking multiple graph convolution mechanisms under the same temporal modelling pipeline, this study systematically analyses the impact of different neighbourhood aggregation strategies on node-level temporal forecasting performance in dynamic graphs.

2.5 Evaluation Metrics

Model performance is evaluated using Mean Squared Error (MSE), which is a standard metric for regression based temporal forecasting tasks. Given a set of node-level prediction $\hat{y}_{i,t}$
And corresponding ground truth $y_{i,t}$ at the time step t, the MSE at time t is defined as

$$MSE_t = \frac{1}{N}\sum_{i=t}^{N} (\hat{y}_{i,t} - y_{i,t})^2 \quad (1)$$

Where N denotes the number of nodes in the graph. The overall test performance is obtained by averaging the MSE across all test time steps T.

$$MSE_{test} = \frac{1}{T}\sum_{t=1}^{T} MSE_t \quad (2)$$

In addition to aggregate error metrics, temporal error profiles are analyzed to assess model stability and robustness over time. By examining the evolution of MSE across successive test snapshots, the ability of each model to adapt to non-stationary temporal dynamic can be evaluated.

2.6 Background

Graph Neural Networks (GNNs) generalize conventional neural architectures by explicitly modelling relational dependencies encoded in graph-structured data through message passing and neighbourhood aggregation. Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the set of vertices and $\mathcal{E}$ represents the set of edges. The fundamental objective of a GNN is to learn latent node embeddings by iteratively aggregating information from each node's local neighborhood. A generic node update operation at layer k can be formulated as:

$$\mathbf{z}_v^{(k+1)} = \phi^{(k)}\left(\sum_{u\in\mathcal{N}(v)} \psi^{(k)}\left(\mathbf{z}_u^{(k)}, \mathbf{z}_v^{(k)}, \mathbf{e}_{uv}\right)\right) \quad (3)$$

where $\mathbf{z}_v^{(k)}$denotes the embedding of node $v$at layer k, $\mathcal{N}(v)$is the neighborhood of v, $\mathbf{e}_{uv}$represents optional edge features, $\psi(\cdot)$is a message function, and $\phi(\cdot)$denotes a learnable transformation followed by a nonlinear activation.

Among early GNN variants, Graph Convolutional Networks (GCNs) introduce a spectral-inspired convolution operation that propagates and smooths node features over the graph topology. The propagation rule of a GCN layer is given by:

$$\mathbf{Z}^{(k+1)} = \sigma\left(\widehat{\mathbf{D}}^{-\frac{1}{2}}\widehat{\mathbf{A}}\widehat{\mathbf{D}}^{-\frac{1}{2}}\mathbf{Z}^{(k)}\mathbf{W}^{(k)}\right) \quad (4)$$

where $\widehat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$denotes the adjacency matrix augmented with self-loops, $\widehat{\mathbf{D}}$is the associated degree matrix, $\mathbf{W}^{(k)}$is a learnable weight matrix, and $\sigma(\cdot)$is a nonlinear activation function.

To improve computational efficiency, Simplified Graph Convolution (SGC) removes intermediate nonlinearities and collapses multiple convolutional layers into a single linear transformation. Instead of stacking several GCN layers, SGC precomputes feature propagation using the normalized adjacency matrix, resulting in:

$$\tilde{\mathbf{Z}} = \left(\widehat{\mathbf{D}}^{-\frac{1}{2}}\widehat{\mathbf{A}}\widehat{\mathbf{D}}^{-\frac{1}{2}}\right)^K \mathbf{X}\mathbf{W} \quad (5)$$

where Kdenotes the propagation depth and $\mathbf{X}$is the input feature matrix. While this formulation significantly reduces training complexity, it may limit expressive power in graphs with complex structural patterns.

Graph Attention Networks (GATs) further enhance message passing by introducing attention mechanisms that assign adaptive importance weights to neighboring nodes. The attention coefficient between nodes $u$and $v$is computed as:

$$\beta_{uv} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^{\top}[\mathbf{W}\mathbf{z}_u \parallel \mathbf{W}\mathbf{z}_v]\right)\right)}{\sum_{k\in\mathcal{N}(v)}\exp\left(\text{LeakyReLU}\left(\mathbf{a}^{\top}[\mathbf{W}\mathbf{z}_k \parallel \mathbf{W}\mathbf{z}_v]\right)\right)} \quad (6)$$

where $\mathbf{a}$ is a learnable attention vector and $\parallel$ denotes vector concatenation. This adaptive weighting mechanism allows GATs to focus on the most informative neighbors, improving representation quality in heterogeneous graphs.

GraphSAGE extends GNNs to an inductive learning setting, enabling generalization to unseen nodes by sampling and aggregating a fixed-size neighborhood. The node update rule can be expressed as:

$$\mathbf{z}_v^{(k+1)} = \sigma\left(\mathbf{W}^{(k)} \cdot \text{AGG}\left(\{\mathbf{z}_u^{(k)} \mid u \in \mathcal{N}(v)\}\right)\right) \quad (7)$$

where AGG(·)denotes an aggregation operator such as mean pooling, max pooling, or LSTM-based aggregation. This inductive formulation enables scalable learning on large and evolving graphs.

Finally, Chebyshev Networks (ChebNet) generalize spectral graph convolutions by approximating graph filters using Chebyshev polynomials of the Laplacian, enabling localized and computationally efficient filtering without explicit eigen-decomposition. The ChebNet convolution operation is defined as:

$$H^{(l+1)} = \sum_{k=0}^{K} T_k(\tilde{L}) H^{(l)} \Theta_k^{(l)} \quad (8)$$

Where $T_k(\cdot)$denotes the Chebyshev polynomial of order k, $\tilde{L}$ is the scaled graph Laplacian, $\Theta_k^{(l)}$are learnable parameters, and Kcontrols the size of the receptive field. By incorporating higher-order neighborhood information, ChebNet effectively captures multi-hop dependencies while maintaining computational efficiency.

2.7 Experimental Setup and Hyperparameter Setting

All experiments were conducted on a workstation equipped with Intel Core I5 CPU, 16 GB of RAM and a Nvidia GPU with 4 GB of memory. The models were implemented in Python using the PyTorch and PyTorch Geometric libraries. To ensure a fair comparison, all GNN models were trained and evaluated under identical hardware conditions, dataset splits and optimization setting. All models shared a common training configuration. The *Adam optimizer* was used with a fixed learning rate, and all models were trained for the same number of epochs using identical batch sizes and activation functions. The hidden embedding dimension was kept constant across architectures to isolate the impact of model design. Model-specific hyperparameters were selected based on standard practices reported in prior studies and lightly adjusted to ensure stable convergence, without extensive hyperparameter tuning. **Table 1** illustrates the main hyperparameter setting used for each model.

Table 1. The model Hyperparameter Setting

Model	Hidden Dim	Learning Rate	Epochs	Batch Size	Model-Specific Setting
GCN	64	0.001	200	32	2 Graph Convolution Layers
GraphSAGE	64	0.001	200	32	Mean Aggregation
GAT	64	0.001	200	32	4 Attention Heads
ChebNet	64	0.001	200	32	*Chebyshev* order K=3
SGC	64	0.001	200	32	Propagation Steps K=2

3. Result and Discussion

This section presents an extensive evaluation of the proposed temporal graph neural network (GNN) models on the WikiMaths datasets. The analysis focuses on training dynamics, temporal generalization, node-level predictive behavior, error characteristics and overall model comparison. Five architectures are evaluated, namely GCN, GraphSAGE, GAT, ChebNet and SGC, each integrated with a recurrent graph convolutional units to model temporal dependencies.

The experimental setup ensures a fair comparison across models by using identical training and testing splits, learning rates, and optimization strategies. Mean Squared Error (MSE) is employed as the primary evaluation metrics, as it effectively captures regression accuracy in a temporal forecasting task. The results are reported through both quantitative metrics and qualitative visualizations to provide comprehensive insight into model behavior.

3.1. Training Loss Analysis

Figure 1 illustrates the training loss curves for all evaluated models over 30 epochs. All models exhibit smooth and monotonic decreases in training MSE, indicating stable optimization and effective learning

dynamics. No signs of divergence or oscillatory behavior are observed, suggesting that the selected learning rate and optimizer are appropriate for the tasks. **Figure 1** was generated from the training loss values recorded at each epoch during the learning process of the GCN, GraphSAGE, GAT, ChebNet, and SGC models. The loss values were computed during training using PyTorch and PyTorch Geometric, and the curves were visualized using Python's Matplotlib library to illustrate the convergence behavior of each model.

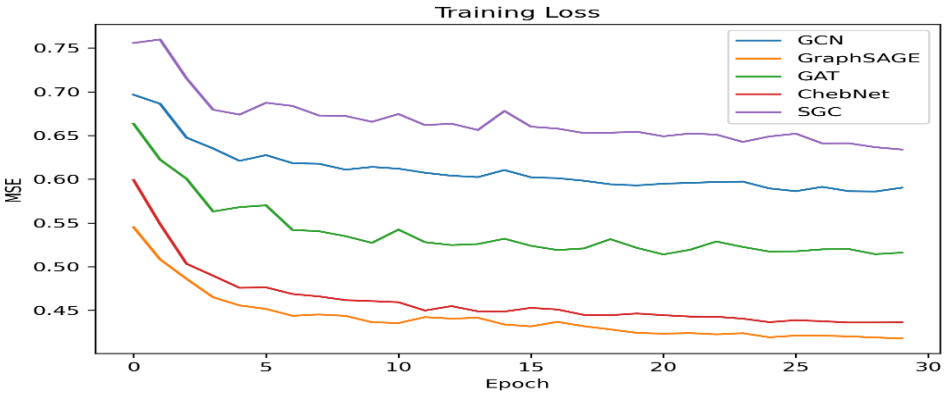

Figure 1. Training Loss analysis

GraphSAGE and ChenNet demonstrate the fastest convergence rates and achieve the lowest final training losses. This behavior indicates their strong capability to capture both local structural information and temporal patterns in the evolving graph. GAT also converges steadily but requires slightly more epochs, which can be attributed to the additional complexity introduced by the attention mechanism.

In contrast, SGC consistently show higher training loss values throughout the learning process. This outcome reflects the limited expressive capacity of simplified graph convolution, particulary in scenarios requiring rich spatial-temporal representations. Overall, the training loss analysis highlights the importance of expressive spatial encoders when combined with temporal recurrence.

4.2. Test MSE Over Time

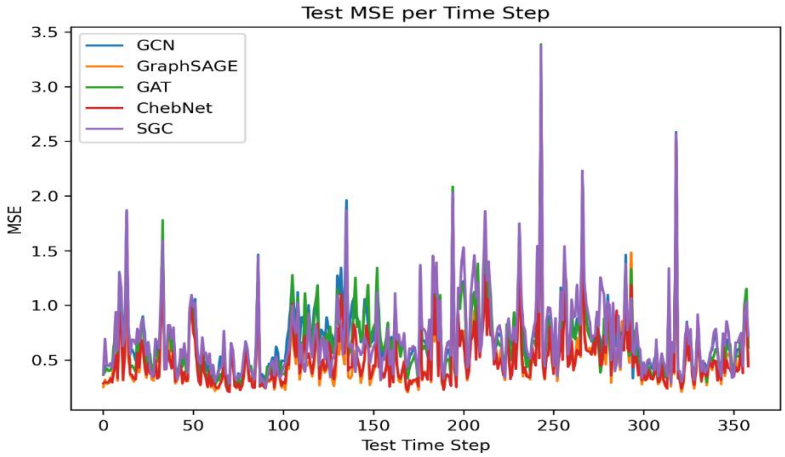

Figure 2. Test MSE Over Time

Figure 2 presents the test MSE computed at each temporal snapshot, providing insight into the models' generalization behaviour over time. All architectures exhibit similar temporal error patterns, with fluctuations corresponding to changes in the underlying dynamics of the WikiMaths interaction graph. These variations indicate that the dataset contains non-stationary temporal behaviour.

GraphSAGE and ChebNet consistently maintain lower MSE values across most time steps, demonstrating robustness to temporal shifts and structural changes in the graph. GAT performs competitively but exhibits higher variance during certain intervals, suggesting sensitivity to abrupt changes in node interactions.

SGC displays the highest error peaks and the largest temporal variability, particularly during periods of rapid change. This result suggests that simplified propagation mechanisms struggle to adapt to dynamic graph environments. Collectively, these findings emphasize the advantage of expressive neighbourhood aggregation for temporal generalization.

Figure 2 was generated from the test Mean Squared Error (MSE) values evaluated at each time step for the GCN, GraphSAGE, GAT, ChebNet, and SGC models. The x-axis represents the test time steps and the y-axis denotes the corresponding MSE values, with the results computed using PyTorch and PyTorch Geometric and visualized using Python's Matplotlib library to illustrate the temporal prediction performance of each model.

4.3. Prediction Vs Ground Truth Analysis

Figure 3 compares the predicted and ground truth values for a representative node across the test horizon. All models successfully capture the overall temporal trend, indicating that the recurrent components effectively model long-term dependencies in the graph signal.

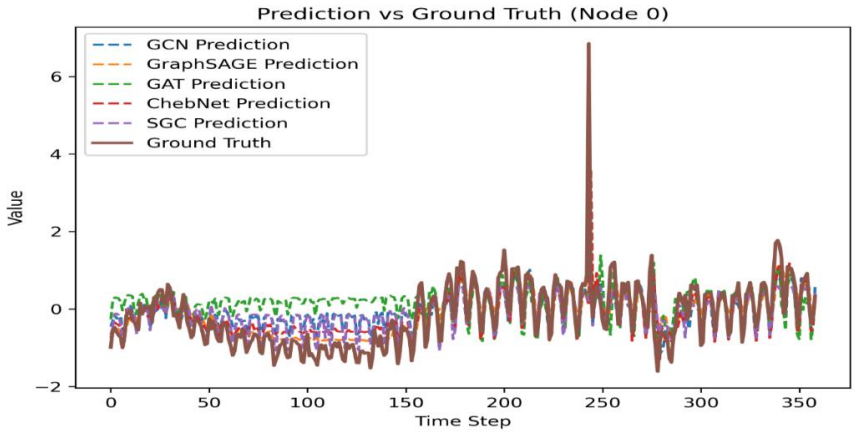

Figure 3. Prediction Vs Ground Truth

GraphSAGE and ChebNet closely follow the ground truth trajectory, including sharp increases and decreases. Their predictions align well with both the magnitude and timing of observed values, highlighting their strong node-level forecasting capability. GAT also performs well but occasionally exhibits minor temporal lag.

In contrast, GCN and SGC tend to produce smoother predictions, leading to underestimation during high-variance periods. This smoothing effect suggests limited responsiveness to sudden changes in node behaviour. The qualitative comparison confirms that richer spatial representations lead to more accurate and responsive predictions.

Figure 3 was generated by plotting the model prediction values and the corresponding ground-truth values at each test time step for the evaluated models. The x-axis represents the time steps, while the y-axis shows the node values ranging from −2 to 6, where the predictions were obtained from the trained models and compared directly against the true target values; all computations were performed using PyTorch and PyTorch Geometric, and the visualization was produced using Python's Matplotlib library.

4.4. Error Distribution Analysis

Figure 4 shows the distribution of prediction errors for each model. The error histograms are centered around zero, indicating that none of the models exhibit systematic prediction bias. This observation confirms that the learned models are well-calibrated.

GraphSAGE and ChebNet produce the narrowest error distributions, reflecting lower variance and more consistent prediction accuracy. The majority of their errors are concentrated near zero, suggesting reliable performance across different temporal contexts. GAT displays a slightly wider distribution but remains competitive.

SGC exhibits the broadest error distribution with heavier tails, corresponding to its larger prediction errors observed in previous analyses. This result further supports the conclusion that simplified graph convolutions are less suitable for complex temporal graph regression tasks.

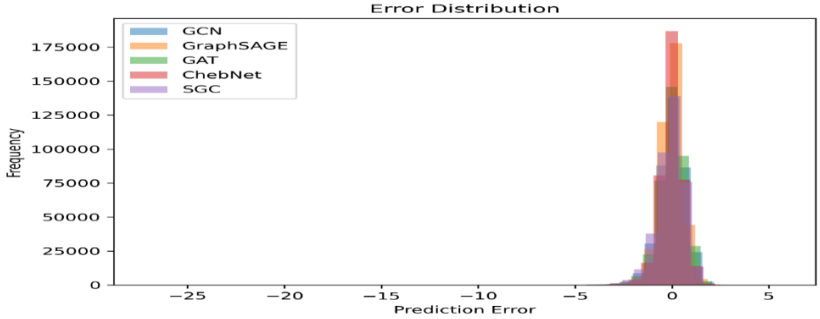

Figure 4. Error Distribution Analysis

Figure 4 was generated from the prediction error values obtained by computing the difference between the model predictions and the corresponding ground-truth values at each test time step. The x-axis represents the prediction error ranging from -25 to 5, while the y-axis shows the frequency of errors, with all error values computed from the experiment which computed using PyTorch and Pytorch geometric. The error distribution is generated from Python Matplotlib library to analyze the spread and bias of prediction errors across models.

4.5. Model Comparison

Figure 5 summarizes the mean test MSE achieved by each model, enabling direct quantitative comparison. GraphSAGE achieves the lowest mean test MSE, followed closely by ChebNet, confirming their superior generalization performance on the WikiMaths dataset.

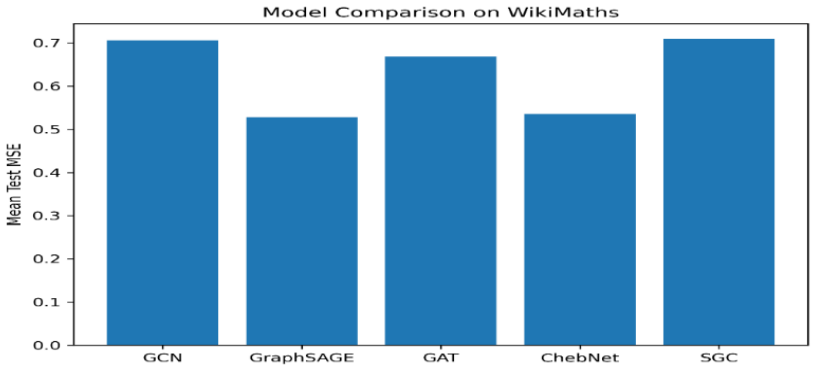

Figure 5. Model Comparison

GAT and GCN achieve intermediate performance levels, balancing model complexity and predictive accuracy. While attention mechanisms enhance expressiveness, they do not consistently outperform inductive neighbourhood aggregation in this setting. These findings suggest that structural inductive bias plays a critical role in temporal graph learning.

SGC yields the highest mean test MSE, reinforcing its limitations in modelling complex spatial-temporal relationships. Overall, the model comparison demonstrates that expressive graph convolution operators are essential for accurate temporal forecasting.

Figure 5 was constructed using the average test Mean Squared Error (MSE) values computed for each model, including GCN, GraphSAGE, GAT, ChebNet, and SGC, over the entire test period. The mean test MSE values were calculated from the model predictions and ground-truth targets using PyTorch and PyTorch Geometric, and the results were presented as a bar chart using Python's Matplotlib library to facilitate a direct performance comparison among the models.

4.6. Training Vs Test Performance

Table 2. Training and Test Performance

Model	Train MSE (Final)	Test MSE (Mean + Std
GCN	0.59 ± 0.01	0.71 ± 0.03
GraphSAGE	**0.42** ± 0.02	**0.53** ± 0.02
GAT	0.51 ± 0.02	0.67 ± 0.03
ChebNet	**0.44** ± 0.01	**0.54** ± 0.02
SGC	0.64 ± 0.01	0.74 ± 0.03

Table 2 summarizes the final training MSE and the average test MSE for each GNN architecture evaluated in this study. All experiments were conducted over 10 independent runs with different random seed, and the number of the table reports the test results correspond to the mean and standard deviation across these runs. As expected, all models exhibit higher error on the test set than on the training set, reflecting the challenge of generalizing to unseen temporal snapshots.

Amont the evaluated models, GraphSAGE and ChebNet demonstrate the smallest discrepancy between training and test performance, achieving test MSE values of 0.53 ± 0.02 and 0.54 ± 0.02 respectively. This limited generalization gap indicates strong robustness and minimal overfitting, suggesting that these architectures effectively capture transferable spatio-temporal patterns. In contrast, SGC exhibits the larger divergence between training and test MSE, with a test error of 0.74 ± 0.03, indicating reduced robustness when exposed to new temporal data. GCN and GAT shows intermediate performance, balancing representation capacity and generalization but with larger performance variability.

4.7 Statistical Significance Test

To evaluate whether the performance differences reported in **Table 2** are statistically significant, this study first examined the distribution of the test MSE values obtained from the 10 independent runs for each model. A normality test was conducted on the test MSE distributions to verify the applicability of parametric statistical testing. Based on this assessment, the distributions were found to be approximately normal, allowing the use of parametric tests for pairwise comparison.

Subsequently, paired t-tests were performed on the test MSE results between models, as all architectures were evaluated on identical data splits and temporal settings. A significance level of $\alpha =$

0.05 was adopted. The analysis indicates that the performance improvements of GraphSAGE and ChebNet over GCN, GAT, and SGC are statistically significant, while the difference between GraphSAGE and ChebNet is not statistically significant, suggesting comparable generalization performance. These results confirm that the observed differences in Table 2 are consistent across runs and not attributable to random initialization effects.

4.8 Discussion

The empirical results consistently demonstrate that the choice of spatial graph convolution operator plays a critical role in temporal forecasting performance, even when the temporal modelling component is held constant. Architectures with expressive neighbourhood aggregation mechanisms, particularly GraphSAGE and ChebNet, achieve superior performance across training dynamics, temporal generalization, and node-level prediction accuracy. These findings align with prior studies suggesting that inductive aggregation [8], [16] and higher-order spectral filtering[12], [20] are effective in capturing heterogeneous structural patterns in dynamic graphs. In contrast, simplified propagation models such as SGC [17], [21], while computationally efficient, exhibit limited capacity to model complex spatial–temporal dependencies, leading to higher error variance and weaker generalization.

The temporal error analysis further highlights the importance of robustness to non-stationary graph dynamics. Models that maintain low and stable test MSE across time steps are better suited for real-world applications where interaction patterns evolve unpredictably. GraphSAGE and ChebNet demonstrate resilience to abrupt changes in graph behaviour, suggesting that their aggregation strategies enable more adaptive representations. Attention-based models such as GAT show competitive performance but exhibit higher temporal variance, potentially due to sensitivity to fluctuating neighbourhood importance weights. This observation indicates that while attention mechanisms enhance expressiveness, they may require careful regularization in highly dynamic settings.

From a broader perspective, these results reinforce the need for benchmarking-oriented studies in temporal graph learning. Many prior works introduce novel architectures without systematically comparing them against strong baselines under controlled conditions. By isolating spatial convolution effects within a unified temporal framework, this study provides clearer empirical evidence of architectural trade-offs that are often obscured in heterogeneous experimental designs. The findings suggest that, for snapshot-based temporal forecasting tasks, the inductive bias introduced by neighbourhood aggregation strategies may be as important as, or even more important than, temporal modelling complexity. This insight has practical implications for model selection and motivates further investigation into the interaction between spatial and temporal components in graph neural networks.

5. Conclusion

This study provides a systematic empirical evaluation of multiple established graph neural network architectures for node-level temporal forecasting on dynamic graphs. The experimental results consistently demonstrate that the choice of spatial graph convolution operator has a significant impact on predictive performance, even when all models share the same temporal modelling mechanism and training configuration. In particular, GraphSAGE and ChebNet exhibit superior generalization across training dynamics, temporal error profiles, and node-level prediction accuracy, while simplified propagation models such as SGC show clear limitations in capturing complex spatial–temporal dependencies.

The primary contribution of this work lies in its benchmarking perspective. This study offers a controlled and reproducible comparison of widely adopted GNN variants under an identical temporal forecasting framework. By isolating architectural effects from confounding factors such as hyperparameter tuning and data preprocessing, the results provide empirical clarity on how different neighbourhood aggregation strategies influence temporal graph learning performance. This contributes to a more transparent understanding of model behaviour that is often obscured in architecture-driven studies.

Despite these contributions, several limitations must be acknowledged. First, the evaluation is conducted on a single temporal graph dataset, which, while large-scale and representative, may not capture the full diversity of structural and temporal patterns observed in other domains. Second, the temporal modelling component is fixed across all experiments, preventing analysis of interactions between different spatial and temporal modelling strategies. Finally, the study focuses on predictive accuracy and does not explicitly evaluate computational efficiency or scalability under varying graph sizes.

Future work may extend this benchmarking framework in several directions. Evaluations on additional temporal graph datasets from different application domains would improve the generalizability of the findings. Exploring alternative temporal modelling mechanisms, such as attention-based or continuous-time approaches, could further clarify the interaction between spatial and temporal components. Additionally, incorporating efficiency and scalability analyses would provide practical guidance for deploying temporal GNNs in real-world systems. Together, these directions offer promising opportunities to deepen empirical understanding in temporal graph learning.

References

[1] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, 2003, pp. 137–146. [online] available at https://dl.acm.org/doi/10.1145/956750.956769

[2] J. Leskovec, M. McGlohon, C. Faloutsos, N. Glance, and M. Hurst, "Patterns of cascading behavior in large blog graphs," in Proceedings of the 2007 SIAM international conference on data mining, SIAM, 2007, pp. 551–556 .

[3] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," IEEE Trans. Knowl. Data Eng., vol. 34, no. 1, pp. 249–270, 2020.

[4] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein, "Temporal graph networks for deep learning on dynamic graphs," arXiv Prepr. arXiv2006.10637, 2020.

[5] S. M. Alzanin and A. M. Azmi, "Detecting rumors in social media: A survey," Procedia Comput. Sci., vol. 142, pp. 294–300, 2018.

[6] S. Kumar, F. Spezzano, V. S. Subrahmanian, and C. Faloutsos, "Edge weight prediction in weighted signed networks," in 2016 IEEE 16th international conference on data mining (ICDM), IEEE, 2016, pp. 221–230.

[7] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in Proceedings of the 24th international conference on world wide web, 2015, pp. 1067–1077.

[8] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," Adv. Neural Inf. Process. Syst., vol. 2017-Decem, pp. 1025–1035, 2017.

[9] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," arXiv Prepr. arXiv1609.02907, 2016.

[10] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," IEEE Trans. neural networks, vol. 20, no. 1, pp. 61–80, 2008.

[11] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in International conference on machine learning, Pmlr, 2017, pp. 1263–1272.

[12] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, "Structured sequence modeling with graph convolutional recurrent networks," in International conference on neural information processing, Springer, 2018, pp. 362–373.

[13] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," arXiv Prepr. arXiv1707.01926, 2017.

[14] H. Han et al., "STGCN: a spatial-temporal aware graph learning method for POI recommendation," in 2020 IEEE International Conference on Data Mining (ICDM), IEEE, 2020, pp. 1052–1057.

[15] A. Pareja et al., "Evolvegcn: Evolving graph convolutional networks for dynamic graphs," in Proceedings of the AAAI conference on artificial intelligence, 2020, pp. 5363–5370.

[16] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan, "Inductive representation learning on temporal graphs," arXiv Prepr. arXiv2002.07962, 2020.

[17] S. M. Kazemi et al., "Representation learning for dynamic graphs: A survey," J. Mach. Learn. Res., vol. 21, no. 70, pp. 1–73, 2020.

[18] B. Rozemberczki et al., "PyTorch Geometric Temporal: Spatiotemporal Signal Processing with Neural Machine Learning Models," in Proceedings of the 30th ACM International Conference on Information and Knowledge Management, 2021, pp. 4564–4573.

[19] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," Stat, vol. 1050, no. 20, pp. 10–48550, 2017.

[20] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering," in Advances in Neural Information Processing Systems, 2016.

[21] F. Wu, T. Zhang, A. H. de Souza, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," in 36th International Conference on Machine Learning, ICML 2019, Pmlr, 2019, pp. 11884–11894.

[22] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," arXiv Prepr. arXiv1709.04875, 2017.