

Adaptable graph region for optimizing performance in dynamic system long-term forecasting via time-aware expert

Received: 21 November 2024

Accepted: 30 September 2025

Published online: 20 November 2025

 Check for updates

Xuan Peng^{1,2}, Zefeng Liu¹, Peng Zhang¹, Yufei Chen¹, Zhanjun Shao¹, Han Zhao¹, Xiaonan Xie¹, Lizhong Jiang¹, Zhuo Huang³, Zhouzhou Pan⁴, Jianwei Yan⁵, Binbin Yin⁶ & Ping Xiang^{1,7} ✉

Real-time and accurate prediction of the long-term behavior of dynamic systems is crucial for identifying risks during unexpected events, while computational efficiency is significantly influenced by the scale of the dynamic system. However, existing neural network models mainly focus on optimizing network structures to improve accuracy, neglecting computational efficiency. To address this issue, we propose regional graph representation, which reduces the scale of the graph structure by merging nodes into region, extracting topological information through graph convolution or lightweight convolution modules, and restoring the regions via fine-grained reconstruction. Notably, this method is adaptable to all graph-based models. Meanwhile, we introduce a sparse time-aware expert module, which selects experts for processing different scale information through a dynamic sparse selection mechanism, enabling multi-scale modeling of temporal information. The architecture we achieve an optimal balance between speed and prediction accuracy, providing a practical solution for real-time forecasting.

Dynamic systems^{1,2} are those whose state evolves over time, continuously shaped by initial conditions, external inputs, and internal laws. The evolution of the system typically follows specific mathematical models or physical laws, which determine its behavior and describe its trajectory. Examples of dynamic systems, such as urban transportation systems³, global atmospheric systems⁴, and train-bridge coupled systems⁵, all exhibit complex structures and highly nonlinear characteristics.

Accurately and rapidly predicting the responses of these dynamic systems is of great significance. Traffic flow is crucial for optimizing urban traffic management, alleviating traffic congestion, and improving travel efficiency. Weather forecasting plays a critical role not only in agricultural production and disaster prevention but also in

monitoring and responding to global climate change. The train-bridge coupled system^{6–8} involves the interaction between trains and bridge structures, with its dynamic characteristics being vital for safety and infrastructure maintenance. Factors such as train weight, speed, bridge load capacity, and environmental conditions^{9–11} all influence the stability of the train-bridge system. Accurately predicting the dynamic behavior of the train-bridge coupled system¹² can effectively prevent bridge structural damage, ensure train safety, and extend the infrastructure's lifespan. Hence, there is an urgent need for real-time, accurate forecasting.

The technologies for modeling dynamic systems¹³ mature considerably, with approach representing dynamic systems as graph structures or grid structure^{14–16} format. In this framework, the system's

¹School of Civil Engineering, Central South University, Changsha, Hunan, China. ²National Engineering Research Center for High-Speed Railway Construction Technology, Changsha, China. ³School of Civil Engineering, Changsha University of Science and Technology, Changsha, Hunan, China. ⁴Department of Engineering Science, University of Oxford, Oxford, UK. ⁵School of Civil Engineering and Architecture, East China Jiaotong University, Nanchang, China. ⁶Department of Civil and Environmental Engineering, The Hong Kong Polytechnic University, Hong Kong, China. ⁷Key Laboratory of Earthquake Resistance and Disaster Prevention of Engineering Structures in Chongqing, Chongqing University, Chongqing, China. ✉e-mail: pxiang2-c@my.cityu.edu.hk

components are treated as nodes, and spatial topological information are incorporated through convolution. However, as the scale of the graph structure increases^{17–19}, certain limitations arise. Specifically, the model's operational efficiency decreases and information exchange becomes difficult.

Recent related works mainly focus on three technical approaches: The first approach is GPU-based sampling methods^{20,21}, which aim to improve sampling efficiency through hardware acceleration. The second approach is structure-aware sampling strategies^{22,23}, where graph clustering is used to identify dense subgraphs and perform block-level sampling, thereby limiting the scope of neighborhood exploration. The third approach involves reducing the number of graph node convolutions^{24–26}, achieving improved computational efficiency while maintaining model performance. Despite various attempts to tackle the challenges of large-scale graphs, existing methods each have limitations: GPU-accelerated sampling only boosts computational speed without addressing the core algorithmic complexity and is highly hardware-dependent; structure-aware sampling can identify dense subgraphs but often requires task-specific strategies, limiting its generality; and reducing the number of convolutional layers can improve efficiency but may compromise the model's ability to capture long-range dependencies, making it difficult to balance efficiency and accuracy.

To address the aforementioned efficiency concerns, we introduce the Regional Graph Representation (RGR) method, which aims to simplify the graph structure while minimizing alterations to the original connectivity. The central concept of RGR is to treat connected nodes as a single region, represented by a new node. The inter-region connections are defined by the links between the nodes within each region. By adjusting the “region order,” the size of each region can be controlled, thereby balancing computational efficiency and accuracy. In conjunction with RGR, we propose the Fine-Grained Reconstruction (FGR) technique, which restores the original number of nodes within each region, enabling the modeling of large-scale dynamic systems. Unlike previous methods, RGR performs coarse-graining on the graph structure, merging connected nodes into regions and applying fine-grained restoration when needed, thereby significantly reducing computational complexity. This compression strategy integrates seamlessly with existing technologies such as GPU acceleration. Through flexible control of the “region order,” the method not only ensures prediction accuracy but also enhances computational efficiency and memory utilization, offering an efficient and adaptive solution for real-time prediction in large-scale dynamic systems.

Another key challenge in long-term time series forecasting lies in balancing the correlation between short-term fluctuations and long-term trends. Multi-scale analysis addresses this by decomposing temporal information across different cycles, enabling the model to process features at each scale in a targeted manner. Currently, two primary temporal modeling techniques are widely utilized in graph neural networks^{27–29}. One involves Conv1D with a gating mechanism³⁰, while the other one utilizes an attention mechanism for temporal information extraction. Both methods effectively address various challenges within the field. However, for dynamic systems characterized by strong periodicity and frequent information exchange, modeling across multiple time scales is particularly important. For instance, in weather systems, temperature fluctuates on both daily and annual cycles. This suggests that analyzing the same temporal data across different time scales can uncover information at different levels.

To perform multi-scale modeling of time series data, we propose a mixture of expert module—sparse time-aware expert. This module comprises multiple expert submodules, each focused on capturing time information at specific time scales, such as short-term fluctuations and long-term trends, by using different convolutional sliding windows. For the input time series, the model dynamically selects and adjusts the most appropriate expert via a router, automatically

determining the optimal time scale and weightedly combining the results from each expert. At the same time, we introduce a shared expert module to ensure the model maintains basic accuracy even when the time scale of the input sequence is not obvious. Compared to traditional methods^{31–33}, the sparse time-aware expert can more accurately capture the dependencies between different time scales, avoiding issues like information loss or excessive smoothing, and overcoming the limitations of single-model approaches in handling multi-scale data.

In graph-based learning, the initial receptive field of a node is typically set to its direct neighborhood, which serves as the core inductive bias of the model. Existing methods for capturing global information have their limitations: the first approach relies on iteratively expanding the receptive field through multiple layers of graph convolution, requiring a large number of stacked layers, which can lead to a bloated model and reduced computational efficiency. The second approach allows nodes to directly access global information (e.g., through global pooling or fully connected layers), which avoids the efficiency issue but can erase the local features of the graph and lose crucial local topological information—this is precisely the core value that distinguishes graph data from other forms of data. To solve the problem of difficult information exchange^{34,35} between distant graph nodes, we propose fusion graph convolution (FGC), which adaptively adjusts the balance between local node information and global information, enabling each node to move beyond reliance on local neighborhood data. By incorporating global information, the method allows nodes to access a broader context, extending beyond their local perspectives.

RGR offers a powerful new way to preprocess large-scale graphs, compressing them by grouping connected nodes while preserving essential structure. This approach can be seamlessly combined with GPU acceleration and integrated into our Regional Graph Neural Network (RGNN) framework. RGNN innovates across graph representation, information interaction, and temporal modeling to enable fast and accurate, end-to-end time series prediction for dynamic systems, laying the groundwork for real-world applications like risk early warning and resource optimization.

Results

The impact of regional graph representation on graph connectivity and computational efficiency

This section investigates the impact of regional graph representation on graph structure size and connectivity under varying region orders. As illustrated in Fig. 1, regional graph representation is applied to the PEMS04 and PEMS08 datasets. The adjacency matrix after regionalization still maintains a high degree of similarity to the adjacency matrix of the original graph structure. This indicates that the regional graph representation preserves the topological information of the original graph structure without destroying it, and also achieves effective preservation of this information. As the region order increases, the graph size gradually decreases. Specifically, the first-order representation reduces the graph size by 50–60%, the second-order by 60–80%, and the third-order compresses the graph to 10–20% of its original size.

To evaluate performance, we measure the model's running speed with a batch size of 64 and compare the results with and without the regional graph representation. Furthermore, we calculate the memory usage of the graph structure processed by the RGR to better understand its contribution to efficiency optimization.

As presented in Table 1, the graph structure utilizing RGR achieves significant reductions in node count, computation time, and memory usage compared to the original graph structure. For the PEMS08 dataset, with region order of 3, the total number of nodes drops from 170 in the original graph to 22, marking an 87% reduction. Concurrently, the computation speed is substantially improved: the

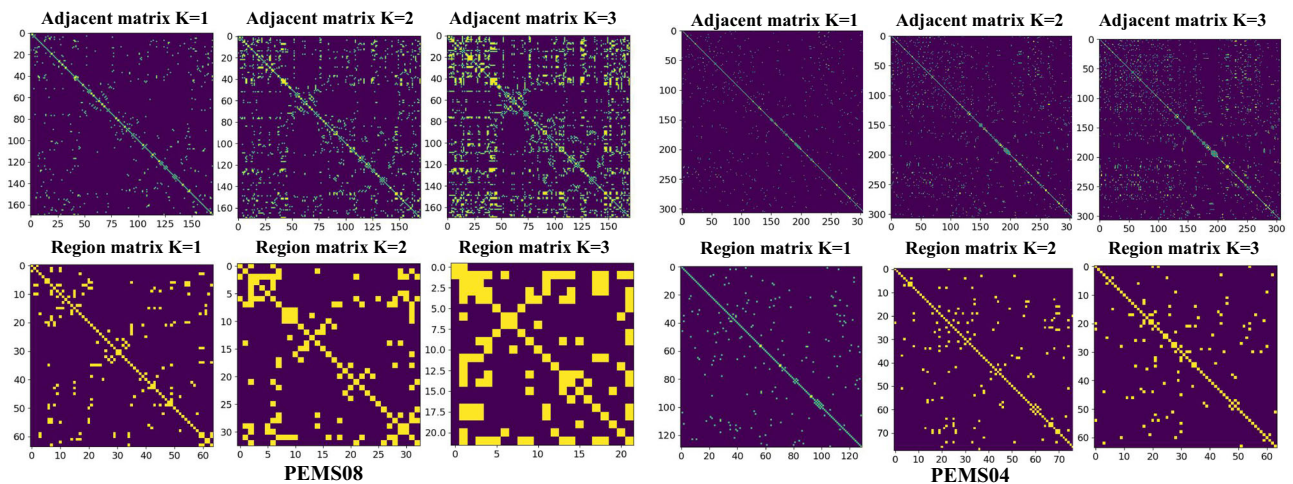


Fig. 1 | The visualization of the regional graph matrix on PEMS04 and PEMS08. The total number of nodes in PEMS08 is 170, and the PEMS04 dataset contains 307 nodes.

Table 1 | Comparison of graph structures with different region orders

Dataset	Graph Structure	Region Order	Total Node Number	Speed	Memory
PEMS08	Original Graph	–	170	1.646 s	23.15 MB
		1	64	0.115 s	8.71 MB
		2	33	0.055 s	4.49 MB
PEMS04	Original Graph	–	307	3.316 s	39.79 MB
		1	129	0.498 s	16.72 MB
		2	78	0.152 s	10.11 MB
		3	64	0.148 s	8.29 MB

processing time decreases from 1.646 s in the original graph to 0.036 s, representing an approximately 45.7-fold acceleration. Memory consumption also sees a notable decrease of 87%, falling from 23.15 MB to 2.99 MB. In the PEMS04 dataset, similar performance gains are observed. When the region order is set to 3, the total node number reduces from 307 to 64, a 79% decrease. The computation speed is enhanced by about 22.4 times, with the processing time shortening from 3.316 s in the original graph to 0.148 seconds. Additionally, memory usage is reduced by 79%, declining from 39.79 MB to 8.29 MB.

Although RGR compresses the graph structure, this compression process is not without losses. As the regional order increases, the finer-grained inter-node connections and local features of the graph structure are inevitably lost. Such information loss may hinder fine-grained analysis tasks, and is particularly noticeable in scenarios that require high-precision node-level predictions.

Predicting the long-term response of the train-bridge coupled system and global climate

Figure 2a demonstrates how to represent a dynamic system in a form that can be understood by neural networks. To validate the generalization ability of RGNN on different graph structures, we used train-bridge coupled system with varying bridge spans and train numbers to predict the train's response under seismic excitation with a hybrid training method as presented in Fig. 2b. At the same time, we tested RGNN's capability to perform real-time predictions on large-scale graph structures via traffic flow dataset. Both tasks are set with a prediction length of 36 steps. Additionally, we designed a global

monthly average temperature time series prediction task, where the model is constructed by inputting a 12-month sequence of multi-dimensional meteorological features, including temperature, humidity, and air pressure, to generate continuous predictions of the monthly average temperature for the following 12 months with a masked training method as presented in Fig. 2c. The detailed results are as shown in Fig. 2d, e.

As presented in Table 2, the RGNN framework consistently outperforms other models (GAT³⁶, STGNN³⁷, and GraphWaveNet³⁸) across various configurations of train numbers and spans, as reflected in the lowest mse and mae values. The results indicate RGNN's strong capability in spatiotemporal prediction, with its optimal performance highlighted in bold.

This advantage highlights RGNN's powerful ability to capture complex spatiotemporal correlations in dynamic systems. This is attributed to the refined spatial modeling capabilities of the RGNN framework, particularly its significant performance in learning spatial structures. Specifically, RGNN effectively extracts and leverages the true topological information from graph structures by integrating graph convolution operations. More importantly, even when the graph structure changes, it maintains efficient extraction capabilities. This flexibility and stability enable RGNN to adapt to structural changes in various dynamic systems, ensuring that it can still accurately capture key relational patterns in complex spatiotemporal environments, thereby enhancing prediction accuracy and reliability.

RGNN also demonstrates excellent performance in temperature prediction tasks, with an average temperature error of only 0.8086° per node. While the overall prediction accuracy is at a high level, there are still relatively significant prediction deviations in some regions. From a technical perspective, the model's high accuracy is mainly attributed to the multi-scale precise modeling capability of the Sparse Time-Aware Expert module—this module can fully capture the complex temporal dependencies in meteorological data, providing reliable support for prediction in the temporal dimension. Meanwhile, the model extracts spatial information from meteorological data in grid structure format through a lightweight convolution module, effectively uncovering the spatial distribution patterns of temperature.

However, the current architecture still has certain limitations in modeling local details, which is the main reason for the prediction deviations in some regions. To address this issue, future work can introduce a residual connection mechanism to fuse the fine-grained local features extracted by shallow convolutions with deep features. This will supplement the model's ability to model meteorological details in local regions (such as microclimate differences caused by

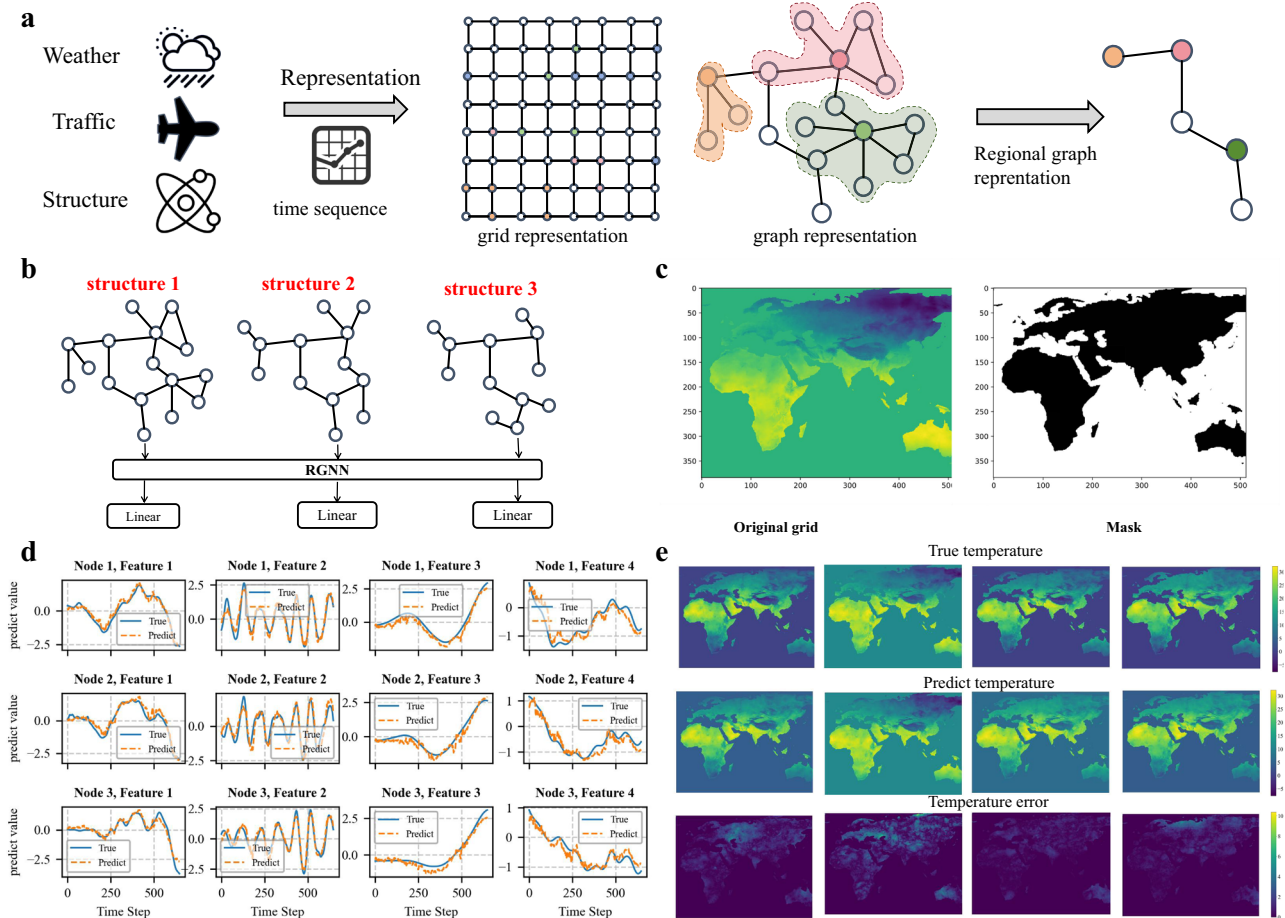


Fig. 2 | Overview of long-term prediction results for dynamic systems.

a Illustration of dynamic systems represented as graph and grid structures. **b** Hybrid training method to enable the model’s applicability to different graph structures. **c** Mask training approach designed for adapting the model to various grid structures. **d** The prediction task in the graph structure focuses on the

response of train nodes in the y and z directions within a train-bridge coupled system under seismic action. Nodes 1 to 4 corresponding to the 1st to 4th carriages respectively, feature 1–4 represent the acceleration of the carriage in the y and z direction. **e** Global monthly average temperature prediction, corresponding to the prediction task in a grid structure.

Table 2 | Comparison of different models under varying numbers of trains and spans

Model	Metric	4 trains			5 trains			6 trains			7 trains		
		5 spans	6 spans	7 spans	5 spans	6 spans	7 spans	5 spans	6 spans	7 spans	5 spans	6 spans	7 spans
GAT ³⁶	MSE	0.418	0.295	0.242	0.353	0.358	0.344	0.350	0.413	0.312	0.392	0.354	0.333
	MAE	0.468	0.376	0.344	0.427	0.423	0.411	0.415	0.447	0.385	0.441	0.421	0.413
STGNN ³⁷	MSE	0.335	0.226	0.184	0.273	0.254	0.263	0.261	0.363	0.229	0.297	0.278	0.317
	MAE	0.407	0.348	0.305	0.373	0.348	0.361	0.342	0.400	0.316	0.374	0.371	0.378
GraphWaveNet ⁴⁹	MSE	0.138	0.103	0.111	0.124	0.140	0.104	0.133	0.159	0.109	0.147	0.154	0.135
	MAE	0.273	0.235	0.235	0.252	0.267	0.233	0.251	0.282	0.232	0.267	0.268	0.255
RGNN	MSE	0.088	0.080	0.072	0.098	0.0991	0.084	0.092	0.122	0.070	0.098	0.107	0.095
	MAE	0.221	0.204	0.189	0.227	0.224	0.209	0.205	0.242	0.186	0.217	0.223	0.211

The predicted length is 36, with the best result presented in **bold**.

terrain, small-scale phenomena like local severe convection, etc.) and further reduce regional prediction deviations.

Ablation experiment of RGNN based on the PEMS dataset

To thoroughly validate the independent roles and collaborative value of each core component in the RGNN framework, we designed and conducted a series of ablation experiments, and the results are shown in Fig. 3. In the experimental design, we first clarified the functional positioning of each component to be validated in the original RGNN

architecture: RGR is responsible for efficiently compressing the original graph structure while retaining core topological information, and FGR is responsible for restoring the region nodes to the target nodes. Together, they form the core feature processing module of RGNN. The spare time aware expert aims to capture dynamic spatiotemporal associative features in the data, while the fusion graph convolution achieves both local and global spatial information modeling.

The prediction target of all ablation experiments is to predict the responses of all nodes within the first region. The first region

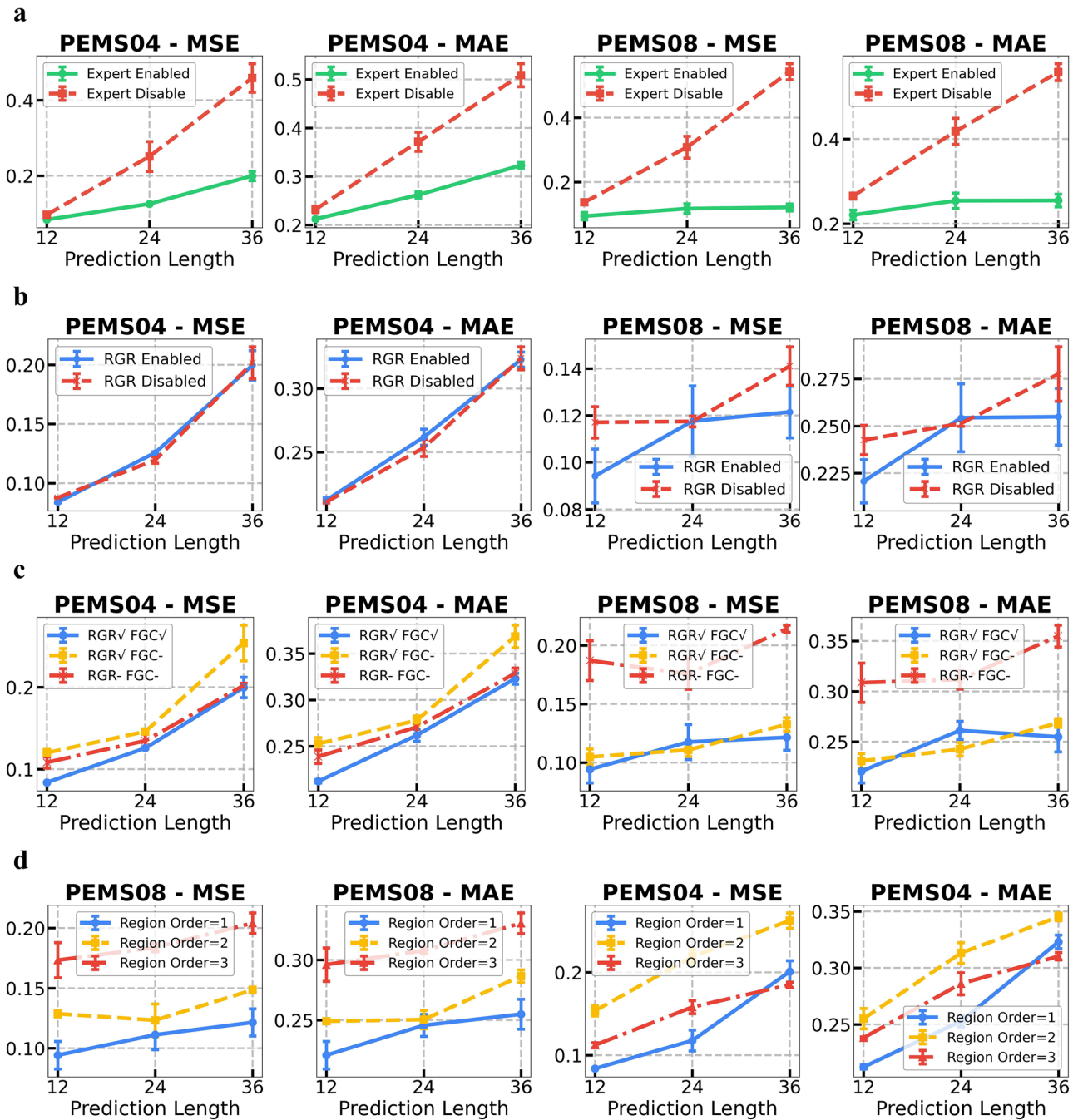


Fig. 3 | The result of the ablation experiment. a Sparse time-aware expert module impact on prediction accuracy and stability. **b** Regional graph representation impact on prediction accuracy and stability. **c** Fusion graph convolution impact on prediction accuracy and stability. **d** Region order impact on prediction accuracy and stability.

represents the area with the most nodes, and selecting this region as the prediction target helps to better evaluate the model's performance on a large number of nodes without sacrificing overall prediction accuracy. This approach allows us to find an ideal balance between improving prediction performance and controlling computational complexity.

Ablation for sparse time aware expert module. Table 3 presents the results of the ablation study comparing the time-aware expert module and the feedforward module, evaluated on two datasets. In the table, “√” indicates that the time-aware expert module is enabled, while “-” indicates that the feedforward module is enabled instead without time-aware expert module.

When the sparse time aware expert module is enabled, the prediction accuracy of RGNN is significantly higher than when using the attention mechanism. This advantage becomes more pronounced as the prediction horizon increases. On the PEMS08 dataset, enabling the time-aware expert module results in a 77.6% reduction in mean squared error (from 0.5432 to 0.1215) and a 54.3% reduction in mean absolute error (from 0.5581 to 0.2549) for a 36-step prediction. Even for the shortest 12-step prediction, the time-aware expert module still brings significant improvements, with MSE and MAE decreasing by 31.3% and 16.9%, respectively. A similar trend is observed on the PEMS04 dataset. For a 36-step prediction, enabling the time-aware expert module reduces MSE by 49.5% and MAE by 29.6% compared to using the feedforward module. For the

Table 3 | Ablation study on time-aware expert, the time-aware expert module is replaced with a basic Feed Forward module to assess its effectiveness

Dataset	Predict Length	RGR	FGC	Time-aware expert	Region Order	MSE	MAE
PEMS04	36	✓	✓	✓	1	0.1996 ± 0.0124	0.3230 ± 0.0059
				-		0.4587 ± 0.0378	0.5089 ± 0.0237
	24	✓	-	✓	-	0.1257 ± 0.0009	0.2619 ± 0.0063
				-		0.2510 ± 0.0397	0.3716 ± 0.0198
	12	✓	-	✓	-	0.0839 ± 0.0006	0.2123 ± 0.0018
				-		0.0971 ± 0.0035	0.2321 ± 0.0065
PEMS08	36	✓	✓	✓	1	0.1215 ± 0.011	0.2549 ± 0.015
				-		0.5432 ± 0.025	0.5581 ± 0.020
	24	✓	-	✓	-	0.1176 ± 0.015	0.2544 ± 0.018
				-		0.3084 ± 0.034	0.4183 ± 0.031
	12	✓	-	✓	-	0.0942 ± 0.0115	0.2207 ± 0.0115
				-		0.1372 ± 0.0005	0.2655 ± 0.007

(The optimal result is **bolded**).

Table 4 | Ablation study on RGR. The regions are restored to the original number of nodes through fine-grained reconstruction

Dataset	Predict Length	RGR	FGC	Time-aware expert	Region Order	MSE	MAE
PEMS04	36	✓	✓	✓	1	0.1996 ± 0.0124	0.3230 ± 0.0059
				-		0.2016 ± 0.0135	0.3239 ± 0.0090
	24	✓	-	✓	-	0.1257 ± 0.0009	0.2619 ± 0.0063
				-		0.1195 ± 0.0026	0.2535 ± 0.0067
	12	✓	-	✓	-	0.0839 ± 0.0006	0.2123 ± 0.0018
				-		0.0877 ± 0.0003	0.2109 ± 0.0008
PEMS08	36	✓	✓	✓	1	0.1215 ± 0.0110	0.2549 ± 0.0150
				-		0.1411 ± 0.0082	0.2776 ± 0.0144
	24	✓	-	✓	-	0.1176 ± 0.0150	0.2544 ± 0.0180
				-		0.1175 ± 0.0023	0.2514 ± 0.0015
	12	✓	-	✓	-	0.0942 ± 0.0115	0.2207 ± 0.0115
				-		0.1171 ± 0.0067	0.2426 ± 0.0078

(The optimal result is **bolded**).

12-step prediction, MSE and MAE are reduced by 13.6% and 8.5%, respectively.

The performance improvement of the sparse time-aware expert module becomes more significant as the prediction length increases. This is primarily due to the module's ability to capture multi-level, multi-scale information in time-series data using convolutional kernels of varying sizes. These kernels process data at different time scales, effectively identifying and extracting complex patterns between short-term and long-term temporal dependencies. Particularly in long-sequence prediction tasks, the different-sized kernels can accurately distinguish patterns with varying periods, thus enhancing the model's ability to understand and forecast long-term trends.

However, when the prediction length is short, the module's multi-scale convolutional kernels are less effective. In short sequences, there is insufficient time to distinguish and extract information at different scales, making it difficult to differentiate between long-term dependencies and short-term fluctuations, which limits the model's performance. As the time series lengthens, the module can uncover more prominent periodic variations and trends over a wider time window, fully leveraging its ability to capture both short-term and long-term dependencies, thereby improving prediction accuracy.

Ablation for regional graph representation and fine-grained reconstruction. Table 4 presents the results of the ablation study on RGR, conducted under the condition that both FGR and the time-aware expert module are enabled. The results indicate that the performance

of RGR is highly dependent on the specific scene. On the PEMS04 dataset, the model performs better without RGR for longer prediction horizons. In contrast, on the PEMS08 dataset, enabling RGR significantly improves the accuracy for prediction lengths of 36 and 12, with negligible differences observed at a length of 24. The maximum accuracy drop does not exceed 13.1%, which is acceptable given the efficiency improvements brought by RGR. These results suggest that the application of RGR is highly feasible, as it not only enhances computational efficiency but also maintains a high level of prediction accuracy in most cases.

The scene dependency of RGR is primarily determined by our regionalization approach. This method introduces varying degrees of distortion to the topological information of different original graph structures, which is the root cause of its scene dependency. Additionally, the accuracy improvement brought by RGR may be related to the scale of the prediction target. Since the prediction target encompasses all nodes within the largest region, the influence of the central node vector may outweigh the effects of simple graph convolution, thereby enhancing prediction accuracy.

It is important to highlight that the main advantage of the RGR method lies in its optimization of runtime efficiency and memory usage. By simplifying the graph structure, RGR significantly reduces computational complexity and memory consumption, as confirmed by prior regional partitioning experiments. Compared to the minor fluctuations in accuracy, this efficiency gain is crucial for large-scale time-series prediction tasks, especially in practical applications requiring

Table 5 | Ablation study on RGR and FGC

Dataset	Predict Length	RGR	FGC	Time-aware expert	Region Order	MSE	MAE
PEMS04	36	✓	-	✓	1	0.2538 ± 0.0217	0.3685 ± 0.0121
		-	-			0.2020 ± 0.0028	0.3286 ± 0.0059
		✓	✓		0.1996 ± 0.0124	0.3230 ± 0.0059	
	24	✓	-		0.1459 ± 0.0031	0.2780 ± 0.0049	
		-	-		0.1347 ± 0.0079	0.2707 ± 0.0101	
		✓	✓		0.1257 ± 0.0009	0.2619 ± 0.0063	
12	12	✓	-		0.1200 ± 0.0042	0.2528 ± 0.0066	
		-	-		0.1083 ± 0.0069	0.2389 ± 0.0073	
		✓	✓		0.0839 ± 0.0006	0.2123 ± 0.0018	
	36	✓	-	✓	1	0.1326 ± 0.0057	0.2687 ± 0.0043
		-	-			0.2140 ± 0.0031	0.3548 ± 0.0108
		✓	✓		0.1215 ± 0.0110	0.2549 ± 0.0150	
PEMS08	24	✓	-		0.1108 ± 0.0055	0.2426 ± 0.0068	
		-	-		0.1756 ± 0.0131	0.3117 ± 0.0094	
		✓	✓		0.1176 ± 0.0150	0.2612 ± 0.0092	
	12	✓	-		0.1048 ± 0.0069	0.2309 ± 0.0075	
		-	-		0.1870 ± 0.0170	0.3086 ± 0.0195	
		✓	✓		0.0942 ± 0.0115	0.2207 ± 0.0115	

We replace FGR with a standard Graph Convolution layer for comparison purposes, which aims to validate the ability of FGR to model large-scale graph structures effectively. (The optimal result is **bolded**).

real-time responsiveness and feasible deployment, where the efficiency improvements provided by RGR far outweigh the slight loss in accuracy.

Ablation for fusion graph convolution. As shown in Table 5, on the PEMS08 dataset, when RGR is not used, the graph structure becomes too large, and relying solely on neighborhood graph convolution for information exchange results in insufficient communication between nodes. This leads to significant shortcomings in the spatial information modeling of the model, causing poor performance of RGNN. However, when fusion graph convolution is enabled, the model's performance is significantly improved. This suggests that Fusion Graph Convolution effectively alleviates the problem of information exchange in large-scale graph structures. By expanding the receptive field of the nodes, the model can better capture the relationships between distant nodes, leading to better spatial modeling performance. This confirms that enhancing the receptive field of nodes is an effective approach.

However, on the PEMS04 dataset, the improvement is not as significant. We believe this is because the PEMS04 dataset already has sufficient global receptive field, and introducing too much redundant information leads to a bottleneck in performance improvement. Excessive redundant information may make the model's learning process more complicated, preventing the model from effectively extracting valuable features, and thus limiting performance gains. This phenomenon indicates that, while enhancing information exchange is typically an important method for improving model performance, in some cases, excessive information exchange and the expansion of the receptive field can have a counterproductive effect, resulting in sub-optimal model performance.

We also find that the combination of RGR and FGC achieves optimal performance in most scenarios. This is particularly interesting, especially when using RGR to reduce the graph structure size, as ordinary neighborhood graph convolution can sufficiently model spatial information without the need for a global receptive field. However, the fact is that using FGC improves the model's performance. We believe this may be because RGR causes some loss of topological information in the graph structure, and FGC compensates for this loss to some extent by weighting the global region node feature vectors.

Ablation for region order. The results presented in Table 6 demonstrate that, across both the PEMS08 and PEMS04 datasets, increasing the region order correlates with a consistent rise in the model's MSE and MAE values, indicating elevated prediction errors. This observation aligns with the model's theoretical design, stemming from two key factors. First, increasing the region order yields coarser regional granularity, which may introduce distortions or redundancies in the topological connectivity between nodes. Second, a larger number of nodes per region amplifies feature dimensional complexity, rendering it more difficult to model and reconstruct the overall regional trends.

In the 36-step prediction for the PEMS04 dataset, Fig. 3 illustrates that when the region order is 3, the model's MSE (0.1850 ± 0.0032) and MAE (0.3103 ± 0.0032) are lower than when the region order is 1 (MSE: 0.2009 ± 0.0132 , MAE: 0.3230 ± 0.0060). This can be explained by the fact that, although increasing the region order complicates reconstruction, the region includes nodes with stable time patterns and low fluctuations, which reduces prediction errors and allows stable feature patterns to dominate the integration process, thereby slightly improving overall prediction accuracy.

Overall, a region order of 1 is optimal for most dynamic system prediction tasks, as it balances accuracy and efficiency and is well-suited for applications with moderate accuracy requirements. When faster prediction speeds are prioritized, increasing the region order can enhance model efficiency: although this may lead to a slight reduction in accuracy, it significantly decreases computational time—particularly when processing large-scale datasets. Thus, selecting an appropriate region order enables a deliberate trade-off between prediction accuracy and computational efficiency, allowing system performance to be optimized according to specific application needs.

Discussion

RGNN is a robust deep learning framework that balances accuracy and efficiency, designed specifically for long-term sequence response prediction in dynamic systems. During the development of deep learning models in the past, many methods often fell into the dilemma of increasing the number of parameters and computational complexity. Although the excessive pursuit of large-scale parameters and high accuracy could enhance the model's expressive power, it also led to a sharp increase in computational complexity, especially when

Table 6 | Ablation study on region order impact

Dataset	Predict Length	RGR	FGC	time-aware expert	Region Order	MSE (Mean \pm Std)	MAE (Mean \pm Std)
PEMS08	36	✓	✓	✓	1	0.1215 \pm 0.0113	0.2549 \pm 0.0124
					2	0.1485 \pm 0.0025	0.2866 \pm 0.0052
					3	0.2041 \pm 0.0085	0.3306 \pm 0.0086
	24				1	0.1112 \pm 0.0124	0.2456 \pm 0.0092
					2	0.1233 \pm 0.0136	0.2504 \pm 0.0076
					3	0.1838 \pm 0.0034	0.3079 \pm 0.0027
	12				1	0.0942 \pm 0.0115	0.2207 \pm 0.0115
					2	0.1287 \pm 0.0026	0.2490 \pm 0.0011
					3	0.1733 \pm 0.0146	0.2962 \pm 0.0140
PEMS04	36	✓	✓	✓	1	0.2009 \pm 0.0132	0.3230 \pm 0.0060
					2	0.2624 \pm 0.0092	0.3451 \pm 0.0035
					3	0.1850 \pm 0.0032	0.3103 \pm 0.0032
	24				1	0.1178 \pm 0.0126	0.2529 \pm 0.0050
					2	0.2190 \pm 0.0087	0.3132 \pm 0.0092
					3	0.1581 \pm 0.0080	0.2860 \pm 0.0097
	12				1	0.0839 \pm 0.0005	0.2123 \pm 0.0018
					2	0.1537 \pm 0.0064	0.2551 \pm 0.0090
					3	0.1122 \pm 0.0031	0.2382 \pm 0.0007

We explore the influence of different region orders on prediction accuracy. (The optimal result is **bolded**).

processing large-scale dynamic systems and long-term sequence data, where the computational cost and storage requirements became increasingly enormous.

We propose a regional graph representation that makes a significant contribution to improving the computational efficiency of graph structures. Specifically, we treat multiple mutually connected nodes as a single region, reducing the number of nodes in the graph. At the same time, the regional design retains the fundamental topological relationships between nodes in the original graph, ensuring the integrity of information transmission and the overall graph structure. This approach allows us to achieve more efficient computational performance without sacrificing the accuracy of the graph. Moreover, as the size of the graph continues to grow, the regional graph representation provides a new solution for large-scale graph computations. It not only holds broad theoretical applications but also brings substantial performance improvements in practical engineering applications. Therefore, this method has significant practical and research value in improving computational efficiency and optimizing resource consumption in graph structures.

To address the issue of insufficient information exchange between nodes in large-scale graph^{34,35}, we propose fusion graph convolution, which adaptively adjusts the balance between local information and global information, which enables each node to balance local and global information, thereby maintaining access to local information while possessing a global receptive field. This approach enhances the comprehensiveness and precision of information flow. This design is particularly effective for handling complex dynamic systems. Although the regional graph representation reduces the size of the graph structure, the graph attention mechanism introduced by fusion graph convolution still leads to some time-consuming operations. Furthermore, the running efficiency is quadratic with respect to the number of nodes. If necessary, the graph attention mechanism can be removed to improve computational efficiency.

Although the framework excels in enhancing efficiency, it still has certain limitations. When predicting nodes across multiple regions, simultaneous information recovery for these regions introduces redundant computational overhead, undermining overall model performance. To mitigate this issue, we propose an enhanced solution: instead of considering all nodes within a region, we focus

exclusively on those that are relevant to the prediction targets. Specifically, we map the dimensions of regional nodes directly to those of the target nodes, thereby eliminating unnecessary computations for irrelevant nodes. This approach resolves the redundancy problem, particularly in cross-region prediction tasks involving large-scale graph.

As presented in Table 3, in time series forecasting tasks—particularly in long-term scenarios with a prediction length of 36—the RGNN architecture integrated with the time-aware expert module demonstrates significantly superior predictive accuracy, outperforming the version without this module by a considerable margin. This comparative result strongly validates the core value of multi-scale temporal analysis in long-term time series forecasting: through the precise capture and modeling of information across different time scales via the time-aware expert module, the model can effectively mine and integrate the implicit multi-cyclic features within time series, thereby maintaining sensitivity to key patterns even in forecasts spanning long temporal length. The time-aware expert module integrates multiple specialized subnetworks—termed sparse experts—each designed to capture temporal patterns at distinct scales. Simultaneous, shared expert module is added to enhance the model's representational capacity. During inference, an adaptive routing mechanism dynamically assigns input subsequences to relevant sparse experts by evaluating temporal patterns embedded in each feature. Finally, the outputs of all expert modules are fused with the output of the shared expert module to generate the final prediction. This process is adaptively adjusted by the modules, with the routing mechanism automatically selecting and optimizing the most suitable expert module for processing based on the input time series.

The RGNN framework is proven to be an effective model that strikes a good balance between speed and accuracy, providing solid technical support for real-time prediction and monitoring of dynamic systems. We plan to apply the RGNN framework to traffic flow management in smart transportation systems, enhancing traffic efficiency and reducing congestion through real-time monitoring and prediction of traffic conditions. Additionally, the framework will also be applied in large-scale structural health monitoring, analyzing and predicting the health status of structures in real-time to help identify potential risks and failures promptly.

Methods

Representation for dynamic system

The representation methods for dynamic systems include two approaches: graph structure and grid structure. Each approach is suitable for different types of systems, and the node features are designed based on the system's characteristics.

In the graph structure representation, the components of the system (such as different subsystems or states) are represented as nodes in the graph, with the relationships or interactions between the nodes represented by edges. The graph structure is suitable for systems with significant interdependencies and connections, such as transportation networks, social networks. In this structure, the features of the nodes typically reflect the system's state or attributes, while the features of the edges represent the interactions or transmission processes between the nodes. Depending on the nature of the system, node features can include physical quantities such as position, velocity, and acceleration, or specific behavior patterns and control parameters.

The grid structure representation is typically used for systems with a regular spatial layout, such as fluid dynamics simulations, weather forecasting, or finite element analysis. In the grid structure, the system is divided into multiple uniform grid cells, with each cell corresponding to a node, and the nodes usually are fixed positions and relationships in space. The grid structure is suitable for dynamic systems that require precise spatial distribution. In this representation, the node features are often related to the spatial distribution of quantities such as position, time, temperature, and pressure, reflecting the system's dynamic changes at different locations or times.

For example, we establish the graph structure of the train-bridge coupled system³⁹. Nodes can be categorized into three types: train nodes, bridge nodes, and pier nodes. Each type possesses distinct physical properties and behaviors, resulting in different feature vectors. Figure 4 illustrates the graph representation of the TBC, where a high-speed train with three carriages travels over a three-span simply supported bridge. Each train carriage is abstracted as a train node, and each span of the bridge is abstracted into one bridge nodes¹⁶. The number of nodes per bridge span unit can be modified; the more nodes there are, the more information the structure contains. Each pier is abstracted as a pier node. Moreover, every train node is interconnected with all bridge nodes; bridge nodes are connected to adjacent bridge nodes; and pier nodes are interconnected with all nodes within their adjacent bridge span units.

For TBC, the x -direction is the driving direction of the train, while y and z are the lateral and vertical directions, respectively. For train nodes, in practical situations, displacement may not always be a feasible option. Generally, the state of a train under seismic influence is

mainly determined by the seismic effects and the train's acceleration. Therefore, even if displacement is used as a feature vector, its impact on the final prediction results is relatively small. So their feature vector consists of displacements and accelerations in the y and z directions.

$$\mathbf{X}_{\text{train}} = \{y_{\text{acc}}, z_{\text{acc}}, y_{\text{dis}}, z_{\text{dis}}\} \in \mathbb{R}^{N_{\text{train}} \times 4}$$

For bridge nodes, real-time responses are not available, so their feature vector

$$\mathbf{X}_{\text{bridge}} = \{L_{\text{span}}, E_{\text{material}}, I_z\} \in \mathbb{R}^{N_{\text{bridge}} \times 3}$$

where L_{span} is the length of each span; E_{material} is the elastic modulus of the material, and I_z is the moment of inertia of the section with respect to the z -axis.

For pier nodes, which serve as input nodes for seismic accelerations, the feature vector

$$\mathbf{X}_{\text{pier}} = \{x_{\text{acc}}, y_{\text{acc}}, z_{\text{acc}}\} \in \mathbb{R}^{N_{\text{pier}} \times 3}$$

consists of seismic accelerations in the x , y , and z directions.

Thus, the graph structure of the TBC is established. The subscripts "acc" and "dis" represent acceleration and displacement, respectively. The prediction target can be flexibly set according to specific task requirements; it can either be the dynamic response of the train (such as operating status, vibration feedback, etc.) or the train's VSI index⁴⁰.

Regional graph representation

The graph structure represents spatial relationships, consisting of nodes and directed edges. Nodes represent system components, and edges demonstrate how information is transmitted between them. Graph structure is represented by the equation:

$$G = (V, E) \tag{1}$$

Where G is the graph, V are the nodes, and E are the edges. An edge between node v_i and node v_j is denoted as e_{ij} , and the neighbors of node v_i are defined as:

$$\mathcal{N}(v_i) = \{v_j \in V; e_{ij} \in E\} \tag{2}$$

The graph structure can be compressed to an adjacency matrix $A \in \mathbb{R}^{n \times n}$, where n is the number of nodes. The adjacency matrix is

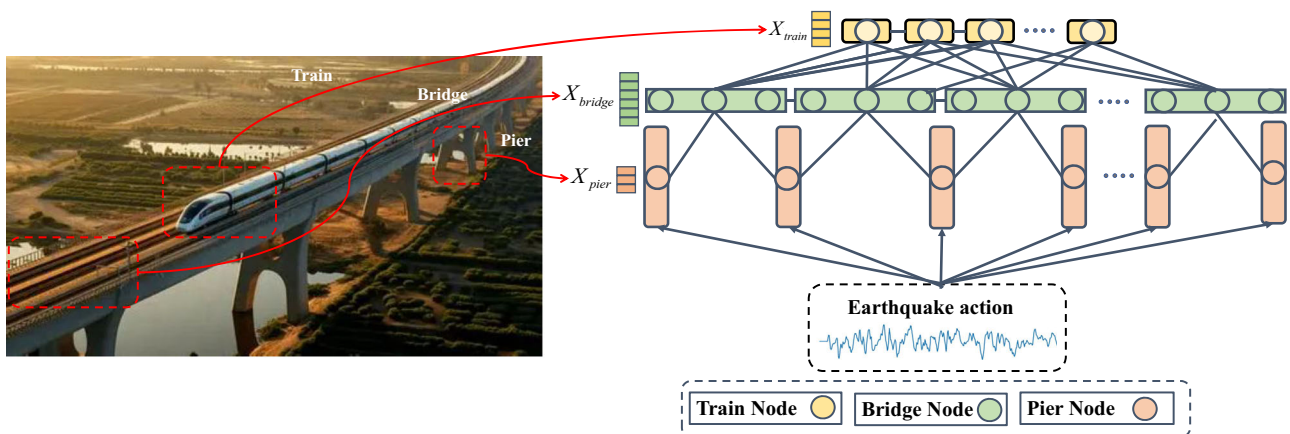


Fig. 4 | Graph representation for train-bridge coupled system. A graph structure is constructed by treating the actual components of the dynamic system as graph nodes and defining the characteristic attributes of these nodes. Notably, the specific attributes of different nodes are determined by the corresponding task requirements.

defined as:

$$A_{ij} = \begin{cases} 1, & \text{if } e_{ij} \in E \\ 0, & \text{if } e_{ij} \notin E \end{cases} \quad (3)$$

This matrix captures the relationships between nodes, representing the graph's topological structure.

To reduce the size of graph structures while preserving essential information, we propose regional graph representation method, as illustrated in Fig. 5. Graph convolution operations inherently introduce an inductive bias, assuming that a node's features are primarily influenced by its neighboring nodes. Based on this assumption, we introduce a strategy: the neighbors of a node are pre-aggregated, with the node itself serving as the central hub of a region, thus consolidating multiple nodes into a single region. In this approach, nodes with the highest connectivity are iteratively identified and assigned as regional centers, while excluding those already regionalized. Through this iterative process, other nodes are progressively aggregated into new regions. To precisely control the scale of regions, we introduce the concept of "region order," as shown in Fig. 6, which defines the number of neighboring layers included within a region. In essence, the region order determines how many levels of neighboring nodes are considered during regionalization. By adaptable adjusting the region order, a flexible balance between information retention and computational efficiency can be achieved. Specifically, the graph structure's order is defined as follows:

$$A_k = \begin{cases} 1, & \text{if } (A^k)_{ij} > 0 \\ 0, & \text{if } (A^k)_{ij} = 0 \end{cases} \quad (4)$$

A^k represents the adjacency matrix using the k -th region order. Where $(A^k)_{ij}$ refers to the element in the i -th row and j -th column of the matrix A^k .

The RGR exhibits two typical extreme cases, both of which effectively validate the rationality of its design logic. The first extreme case is a graph where all nodes are disconnected. Since such a graph lacks meaningful topological relationships, the RGR will leave the original graph structure unchanged during processing, neither forcibly merging nodes nor adding additional edges. The second extreme case is a fully connected graph. In this case, the RGR will merge all nodes in the graph into a single regional node, significantly simplifying the original graph structure. This simplification aligns perfectly with our design goal of—reducing the graph size through region merging. It essentially performs a highly efficient graph pooling process, preserving the global information of the entire graph while minimizing computational complexity. The behavior in these two extreme scenarios fully supports the correctness of the RGR design rationale: it neither disrupts the original state of graphs without topological relationships nor fails to simplify highly connected graphs effectively.

The regional graph representation compresses the graph structure, but still face a limitation: when node-level predictions are needed, individual node features are merged into a central node representation, hindering detailed analysis. To address this, we introduce the fine-grained reconstruction mechanism, which redistributes the central node's feature vector to all constituent nodes in the region, optimizing the use of information.

The fine-grained reconstruction process is described by:

$$v_{r'} = \text{reshape}(\mathbf{W}_r \cdot v_r) \quad (5)$$

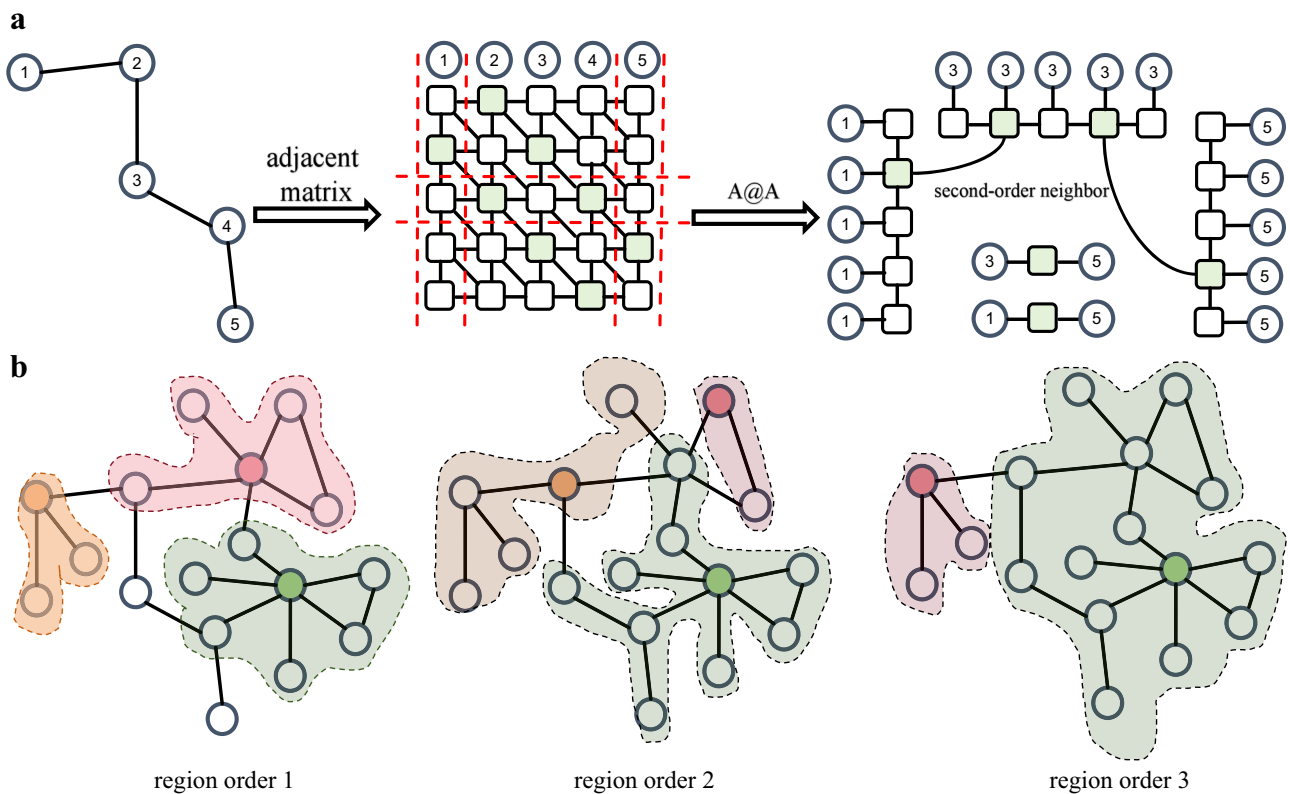


Fig. 5 | Detailed explanation of region order. a The size of the region is controlled by adjusting the order of adjacent nodes of the central node (i.e., the order of neighbors). The figure illustrates the connection path of node 1: node 1 first associates with node 3, and then connects to node 5 through node 3, forming the

second-order neighbor relationship of node 1. **b** Different region orders correspond to different sizes of the regionalized graph. As the order increases, the size of the graph region grows, incorporating more neighboring nodes.

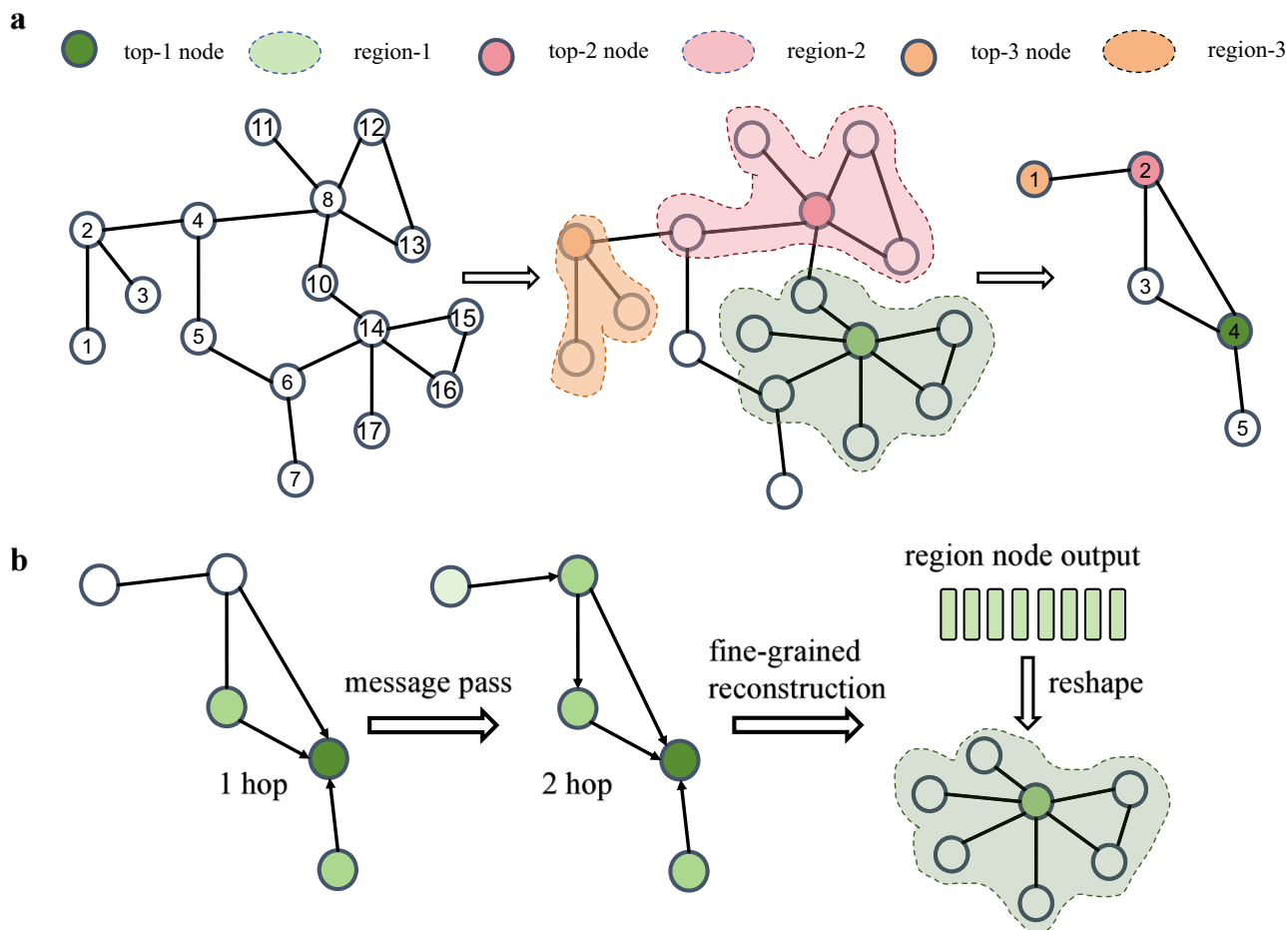


Fig. 6 | Specific implementation of regional graph representation and fine-grained reconstruction. **a** Visualization of regional graph representation. The graph is divided into three independent regions. The original structure, initially with 16 nodes, is simplified by merging redundant nodes based on regionalization. After the process, the node count reduces to 5. **b** Fine-grained reconstruction

method. After information interaction, regional nodes are projected into a high-dimensional space, where potential associations are preserved via feature mapping. A reshape operation is then applied to restore the complete nodes in the region, achieving precise reconstruction from simplified regional nodes to the original graph.

Where $v'_r \in \mathbb{R}^{n_r \times d}$ is the feature vector of the nodes in the region, $\mathbf{W}_r \in \mathbb{R}^{d_r \times (n_r \times d)}$ is the linear transformation matrix, and $\mathbf{v}_r \in \mathbb{R}^{d_r}$ is the feature vector of the center of the region. Here, d is the feature dimension of the nodes within the region; d_r is the feature dimension of the region center, and n_r is the number of nodes need be restored.

The fine-grained reconstruction mechanism, via refined feature mapping and node-level information decoding, accurately recovers individual node features compressed during regional aggregation. To address feature loss issues—such as blurred local details and attenuation of critical attributes—that may arise from regionalization operations, the mechanism constructs a reverse mapping pathway from regional global features to node-level local features, thereby effectively mitigating prediction biases induced by feature loss. On one hand, it preserves the enhanced computational efficiency afforded by regional graph representation through node aggregation and scale reduction, anchoring the model framework consistently in simplification. On the other hand, by prioritizing recovery of node features critical to prediction tasks, it enhances the targeting of detailed information restoration. This “simplification as foundation, restoration as application” design philosophy ultimately achieves a dynamic balance between computational efficiency and feature integrity: it avoids redundant computations in the original graph structure while overcoming information loss from excessive regionalization, thus providing technical support integrating efficiency and accuracy for precise prediction of complex dynamic systems.

The details of the regional graph representation algorithm are described in Algorithm. 1.

Algorithm 1. Regional graph representation for graph structure
Input: Original adjacency matrix A ; Region order k ; feature vector x_v

Output: Adjacency matrix A_r of regional graph structure; Region feature vector x_{v_r}

1: Initialize A_k (as defined in equation (4)), and initialize sets \mathcal{R} and \mathcal{E} to record region centers and edges, respectively.

2: **While** sum (A_k) $\neq 0$ **do**

3: Find the node v with the highest degree and add v to \mathcal{R} ;

4:

$$v = \arg \max_i \left(\sum_j A_{k_{ij}} \right)$$

5: Select node v as the center of the region

6: Find the k -hop neighbors of node v ;

7:

$$N_k(v) = \{i : d(i, v) \leq k\}$$

8: Compute the feature vector for the region:

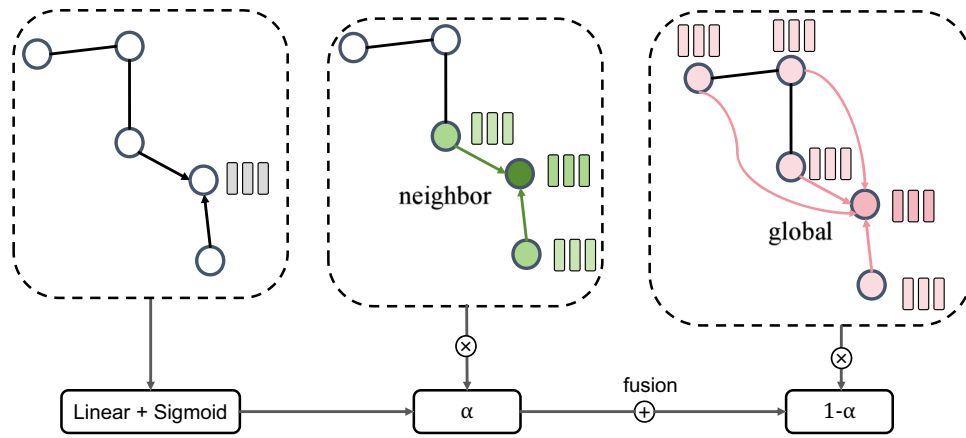


Fig. 7 | The details of fusion graph convolution. fusion graph convolution employs a gate mechanism to dynamically adjust the influence weights of each node on both local (neighbor) and global information within the graph structure, facilitating adaptive selection of diverse information sources.

9:

$$x_{v_r,i} = \frac{1}{|N_k(v)|} \sum_{i \in N_k(v)} x_{v_i}$$

10: Remove the region nodes from A_k ;

11:

$$A_k = A_k - A_k[N_k(v), N_k(v)]$$

12: **end while**

13: Establish connectivity between regional center nodes

14: **for** each pair $r_i, r_j \in \mathcal{R}$ **do**

15: **if** $\exists u \in r_i, \exists v \in r_j$ such that $A_{u,v} \neq 0$ **then**

16: $\mathcal{E} \leftarrow \mathcal{E} \cup \{(r_1, r_2)\}$

17: **end if**

18: **end for**

19:

$$A'_{i,j} \leftarrow \begin{cases} 1 & \text{if } (r_i, r_j) \in \mathcal{E}, \\ 0 & \text{otherwise.} \end{cases}$$

20: $x_{v_r} \leftarrow \{x_{v_i} | i \in \mathcal{R}\}$

Fusion graph convolution

We introduce fusion graph convolution (shown in Fig. 7), which integrates both local and global information in graph structure. This method utilizes a gating mechanism to selectively regulate the flow of local (neighbor) and global information at the node level.

For global information aggregation, node-wise information exchange is performed via an attention mechanism⁴¹. For each pair of nodes (v_i, v_j) in the graph, a learnable attention mechanism is employed to calculate the importance of node v_i to node v_j , as shown in the following formula:

$$\alpha_{ij}^t = \frac{\exp(\text{LeakyReLU}(\mathbf{a}^\top [Wx_i^t | Wx_j^t]))}{\sum_{k \in \mathcal{N}(i)} \exp(\text{LeakyReLU}(\mathbf{a}^\top [Wx_i^t | Wx_k^t]))} \quad (6)$$

where \mathbf{a}^\top is a learnable attention weight vector, '||' denotes concatenation, W is a feature transformation matrix, and α_{ij}^t is the attention weight of node v_j to v_i (normalized via softmax).

Node features are aggregated using these attention weights, with neighbors defined as all nodes in the graph to enable global

information propagation:

$$h_i^t = \sigma \left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij}^t Wx_j^t \right) \quad (7)$$

Multi-head attention⁴² enhances expressiveness by performing the aggregation K times with distinct attention heads, then concatenating results:

$$h_i^t = \parallel_{k=1}^K \sigma \left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij}^{t,k} W_k x_j^t \right) \quad (8)$$

where k indicates the k -th attention head.

To balance global and local information, fusion graph convolution is defined as:

$$\begin{aligned} \beta_i^t &= \sigma(W_\beta x_i^t + b_\beta) \\ h_i^t &= (1 - \beta_i^t) \sigma \left(\sum_{j \in \text{all}} W_{\text{global}} x_j^t \right) + \beta_i^t \sigma \left(\sum_{j \in \mathcal{N}(i)} W_{\text{local}} x_j^t \right) \end{aligned} \quad (9)$$

where $W_\beta, W_{\text{global}}, W_{\text{local}}$ are linear transformation matrices; t is the time step; "all" denotes all graph nodes; $\mathcal{N}(i)$ is the local neighborhood of node i ; and β_i^t (gating coefficient) weights the fusion of global and local features.

Detail module in RGNN architecture

As shown in Fig. 8, RGNN adopts an encoder-only architecture and is trained via a one-step prediction approach based on supervised learning. Specifically, before the data is processed by RGNN, we abstract the dynamic system into either a graph structure or a grid structure, depending on the differences in the dynamic system's structure. We then perform feature embedding for each node and simultaneously upsample the features of the embedded nodes, ensuring that all nodes have the same feature dimension.

The feature fusion component of RGNN specifically consists of two parts: temporal information extraction and spatial information extraction. Specifically, temporal information extraction is controlled by the sparse time-aware expert module, which contains several distinct temporal experts. Only a subset of these experts is activated at each time, ensuring operational efficiency. Finally, by selecting

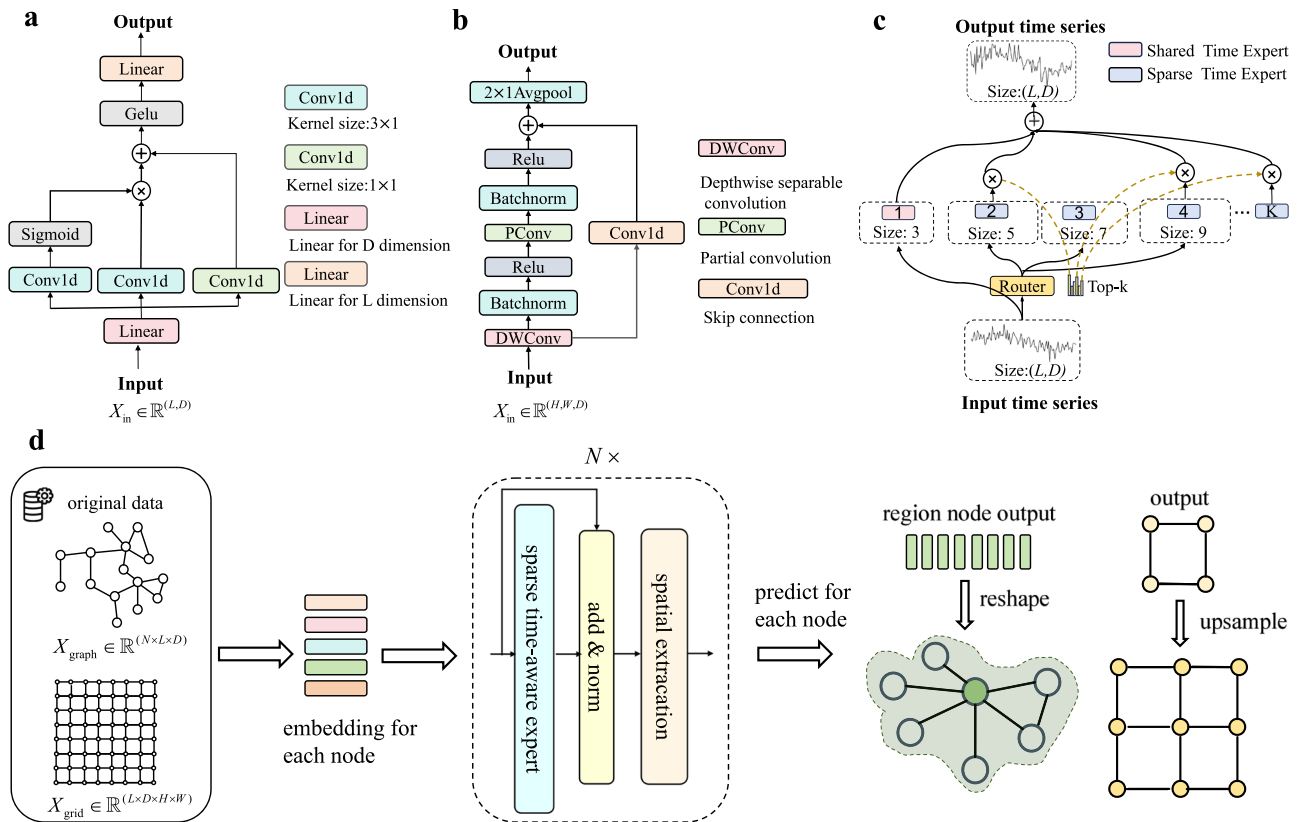


Fig. 8 | The detail module in the RGNN architecture. **a** The operational logic of the temporal expert module, which uses Conv1D to extract temporal information, followed by a gating mechanism. A linear layer is then employed to integrate multiple feature inputs. **b** Detailed implementation of the convolution block for the grid structure. This is achieved by designing depthwise separable convolutions and partial convolution modules to reduce computational load and improve processing speed. **c** Sparse time-aware expert module. The router allocates the input temporal

information to different time experts, each processing temporal information at various scales. The results from all temporal experts are then weighted and fused. **d** The operational logic of the RGNN framework: The input dynamic system is transformed into graph or grid-formatted data. The input is fed into the RGNN architecture for feature extraction. Then it is passed through fine-grained reconstruction or upsampling to reconstruct all nodes.

different experts and fusing their results, multi-scale modeling of temporal information is achieved.

Each expert model is specifically dedicated to processing information at a distinct scale, thereby enabling multi-scale modeling. Specifically, we achieve targeted extraction of scale-specific information by adjusting the kernel size of the Conv1d layer within each expert module. From a technical perspective, linear layers are appended to both the input end (head) and output end (tail) of each expert module, which are respectively utilized for feature dimension-wise information projection and temporal dimension-wise information aggregation. Within the expert module, a gating mechanism is employed: first, a convolution operation is performed on the input temporal features along the temporal length dimension; subsequently, the convolution output is subjected to gating weighting via a convolution operation from an auxiliary branch, followed by a sigmoid activation function. The specific formulas are as follows:

$$Z_{in} = \text{Linear}_D(Z_{in}) \quad Z_{weight} = \text{Conv1D}_{i,2}(X_{in}, K_i) \quad (10)$$

$$Z_{hidden} = \text{Conv1D}_{i,1}(X_{in}, K_i) \quad Z_{gate} = Z_{hidden} \odot Z_{weight} \quad (11)$$

$$Z_{out} = Z_{gate} + \text{Conv1D}_{i,3}(X_{in}, \mathbf{1}) \quad Z_{out} = \text{Linear}_L(Z_{out}) \quad (12)$$

Here, Linear_D and Linear_L denote linear layers for the feature dimension and temporal dimension, respectively, which are used to perform linear mapping on the feature information of the

corresponding dimensions. For $\text{Conv1D}_{i,j}$ (where $j \in \{1, 2, 3\}$), the subscript i represents the i -th time-aware expert, and the subscript j corresponds to the j -th Conv1D operation shown in Fig. 8a.

In terms of input, $X_{in} \in \mathbb{R}^{N \times L \times D}$ represents the time series information input to the time-aware expert module. N is the number of nodes, L is the temporal length, and D is the feature dimension. For the output, $X_{out} \in \mathbb{R}^{N \times L \times D}$ denotes the final output after processing by the time-aware expert module.

For the input time series, we adopt a sparse mechanism to extract information from the temporal dimension. The specific implementation process is as follows: First, a Router calculates the weight scores of each temporal expert, and the top- k experts are selected to participate in subsequent multi-scale feature processing—where k is a hyperparameter, whose value directly controls the number of currently activated expert modules, thereby realizing the sparsity regulation of the model. Meanwhile, to ensure the stability of basic temporal information processing capabilities, we additionally set up a shared time-aware expert to serve as a supplement to the core processing modules. The specific formulas for the relevant calculation process are as follows:

$$s = \text{Router}(X_{in}) \in \mathbb{R}^N \quad (13)$$

$$g_i = \begin{cases} s_i & \text{if } i \in \text{Topk}, (k < N) \\ 0 & \text{otherwise} \end{cases} \quad (14)$$

$$g'_i = \frac{g_i}{\sum_{j=1}^k g_j} \quad (15)$$

$$X_{\text{spare}} = \sum_{i=1}^k g'_i \cdot \text{TimeExpert}^{(i)}(X_{\text{down}}^i) \quad (16)$$

$$X_{\text{share}} = \text{TimeExpert}(X_{\text{in}}) \quad (17)$$

$$X_{\text{out}} = X_{\text{spare}} + X_{\text{share}} \quad (18)$$

Since the spatial structure of the input dynamic system exists in two distinct forms—graph structure and grid structure—we design two targeted spatial information extraction schemes to adapt to this difference. Specifically, for graph-structured data, we adopt the method defined in Eq. (9) for spatial feature extraction; for grid-structured data, we realize the effective capture of spatial information through the module illustrated in Fig. 8b. Specifically, we employ a lightweight convolution design by combining depthwise convolution with partial convolution.

RGNN introduces the sparse time-aware expert module for multi-scale fusion. After the multi-scale fusion, RGNN proceeds with spatial information extraction. Different methods are used to extract spatial information depending on the data structure. For graph-structured data, fusion graph convolution is employed to extract spatial information. For grid-structured data, convblock (as illustrated in Fig. 8) is used for spatial information extraction. This process is repeated N times for the extraction and fusion of temporal and spatial information, ultimately outputting the condensed result. Finally, for graph-structured data, the Fine-grained Reconstruction method is applied to restore the details of regional nodes; for grid-structured data, Upsampling Fusion Convolution is used to restore the data to its original size.

Training setting

In order to ensure consistency in model configurations across different datasets and enable fair comparisons, a unified experimental protocol is employed. All models and modules are implemented using PyTorch to maintain consistency in implementation. Regarding dataset partitioning and normalization, the datasets are split into training, validation, and test sets with proportions of 60%, 30%, and 10%, respectively. Only the training set is used for normalizing the validation and test sets.

For the optimizer and learning rate, the Adam optimizer is chosen with a default learning rate of $3e-3$, and the learning rate is decayed by a factor of 0.8 every 5 epochs during training. In terms of model training, a triple independent training scheme is adopted, where each model configuration is trained from scratch three times with different random initializations. Early stopping is implemented during training. Two checkpoints are saved for each training run: one corresponding to the model with the lowest validation loss, and the other corresponding to the final model at the end of training. The checkpoint yielding the best performance on the test set is selected as the representative result for that configuration. Model performance is evaluated using two standard metrics: Mean Squared Error (MSE) and Mean Absolute Error (MAE).

To balance computational efficiency, we set the number of layers in the RGNN to 2. Additionally, when using sparse time-aware experts, we set five time aware experts by default, with a sparsity coefficient k set to 3. At the same time, we default to using regional graph representations for all models, with the default region order set to 1.

Training method. To enable the RGNN architecture to better adapt to dynamic systems with diverse structures, two innovative training

methods is introduced as shown in Fig. 2: the hybrid training method⁴³ and the masked training method. These methods are optimized for graph and grid structures, respectively. The combination of these approaches aims to improve the RGNN architecture's performance across various complex datasets and enhance its generalization capabilities.

Specifically, the hybrid training method integrates the training processes of multiple graph structures, designing a dedicated linear layer for each structure to output the corresponding node results. Despite structural differences, the core architecture of RGNN remains consistent, ensuring scalability and efficient utilization of graph information. By sharing the RGNN module, this method eliminates the high cost of training each graph structure individually. In contrast, the masked training method employs a masking mechanism, ensuring that the model focuses only on unmasked grid nodes, thereby ignoring irrelevant nodes. This approach not only reduces computational resource waste but also improves training efficiency and accuracy when handling diverse grid structures. Consequently, the model can simultaneously train on different grid data, enhancing its adaptability.

Dataset. In the research framework and experimental verification process of this paper, the three core datasets involved each possess distinct and unique intrinsic attributes and structural characteristics. These characteristics are not only reflected in basic aspects such as data sources, scale, and content dimensions but also profoundly influence data quality, usability, and applicable scenarios.

The TBC dataset, obtained from Matlab computations^{44,45}, exhibits negligible discrepancies when compared to structural simulations. Therefore, future research will prioritize data from numerical computations⁴⁶, using seismic waveforms based on the Clough-Penzien model⁴⁵ and recorded by high-precision sensors during actual earthquake events.

CRU-TS⁴⁷ dataset, includes key climate variables like average minimum and maximum temperatures (°C) and total precipitation (mm). It is essential for studying global climate dynamics, regional weather patterns, and climate change impacts. The dataset spans 384 longitudes, 512 latitudes, and covers 360 months of monthly data.

The PEMS⁴⁸ dataset comprises data from multiple sensors across different highways in California, used for real-time monitoring of traffic flow, vehicle speed, lane occupancy, and other traffic information. Each sensor records data such as traffic flow, average speed, and occupancy rate.

Data availability

The supporting data for this study comes from open-source datasets, and the relevant data are cited within the paper. The train-bridge coupled system dataset will be open-sourced in the future. For further details and access to the datasets, please refer to the UCI Machine Learning Repository at <https://archive.ics.uci.edu/>.

Code availability

The full implementation code for the RGNN framework can be found at the following link: <https://github.com/Amos-Yooupi/RGNN>. This repository includes the complete model implementation, along with the associated training and evaluation scripts, making it easier for researchers and developers to reproduce and further optimize the framework. Additionally, the code for all comparison models has also been open-sourced. Users can access and use this code for comparative experiments, thereby gaining a deeper understanding of the performance of the RGNN framework across different tasks and datasets.

References

- Jolly, W. M. et al. Climate-induced variations in global wildfire danger from 1979 to 2013. *Nat. Commun.* **6**, 7537 (2015).

2. Avila, A. M. & Mezić, I. Data-driven analysis and forecasting of highway traffic dynamics. *Nat. Commun.* **11**, 2090 (2020).
3. Mittal, K. M., Timme, M. & Schröder, M. Efficient self-organization of informal public transport networks. *Nat. Commun.* **15**, 4910 (2024).
4. Liang, Y.-C. et al. Amplified seasonal cycle in hydroclimate over the Amazon River basin and its plume region. *Nat. Commun.* **11**, 4390 (2020).
5. Ma, L., Li, Z., Xu, H. & Cai, C. S. Numerical study on the dynamic amplification factors of highway continuous beam bridges under the action of vehicle fleets. *Eng. Struct.* **304**, 117638 (2024).
6. Zhai, W., Han, Z., Chen, Z., Ling, L. & Zhu, S. Train-track-bridge dynamic interaction: a state-of-the-art review. *Veh. Syst. Dyn.* **57**, 984–1027 (2019).
7. Zhai, W., Wang, K. & Lin, J. Modelling and experiment of railway ballast vibrations. *J. Sound Vib.* **270**, 673–683 (2004).
8. Zhai, W., Wang, K. & Cai, C. Fundamentals of vehicle-track coupled dynamics. *Veh. Syst. Dyn.* **47**, 1349–1376 (2009).
9. Zeng, Q., Yang, Y. & Dimitrakopoulos, E. G. Dynamic response of high speed vehicles and sustaining curved bridges under conditions of resonance. *Eng. Struct.* **114**, 61–74 (2016).
10. Du, X. T., Xu, Y. L. & Xia, H. Dynamic interaction of bridge train system under non-uniform seismic ground motion. *Earthq. Eng. Struct. Dyn.* **41**, 139–157 (2012).
11. Homaei, H., Stoura, C. D. & Dimitrakopoulos, E. G. Extended modified bridge system (EMBS) method for decoupling seismic vehicle-bridge interaction. *Earthq. Eng. Struct. Dyn.* **53**, 4054–4075 (2024).
12. Zhao, H. et al. Safety analysis of high-speed trains on bridges under earthquakes using an LSTM-RNN-based surrogate model. *Comput. Struct.* **294**, 107274 (2024).
13. Fay, D. & Ringwood, J. V. On the influence of weather forecast errors in short-term load forecasting models. *IEEE Trans. Power Syst.* **25**, 1751–1758 (2010).
14. Cheng, G., Gong, X.-G. & Yin, W.-J. Crystal structure prediction by combining graph network and optimization algorithm. *Nat. Commun.* **13**, 1492 (2022).
15. Meller, A. et al. Predicting locations of cryptic pockets from single protein structures using the pocketminer graph neural network. *Nat. Commun.* **14**, 1177 (2023).
16. Xiang, P. et al. Adaptive gn block-based model for seismic response prediction of train-bridge coupled systems. *Structures* **66**, 106822 (2024).
17. Batarfi, O. et al. Large scale graph processing systems: survey and an experimental evaluation. *Clust. Comput* **18**, 1189–1213 (2015).
18. Zhao, X., Liang, J. & Wang, J. A community detection algorithm based on graph compression for large-scale social networks. *Inf. Sci.* **551**, 358–372 (2021).
19. Kuo, P.-C. et al. Gnn-lstm-based fusion model for structural dynamic responses prediction. *Eng. Struct.* **306**, 117733 (2024).
20. Sun, J. et al. Legion: automatically pushing the envelope of Multi-GPU system for Billion-Scale GNN training. In *Proc. USENIX Annual Technical Conference (USENIX ATC 23)*, 165–179. <https://www.usenix.org/conference/atc23/presentation/sun> (USENIX Association, 2023)
21. Yang, J. et al. Gnnlab: a factored system for sample-based gnn training over gpus. EuroSys'22, 417–434. <https://doi.org/10.1145/3492321.3519557> (Association for Computing Machinery, 2022).
22. Ying, R. et al. Graph convolutional neural networks for web-scale recommender systems. KDD'18, 974–983. <https://doi.org/10.1145/3219819.3219890> (Association for Computing Machinery, 2018).
23. Chiang, W.-L. et al. Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks. In *Proc. 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. <https://api.semanticscholar.org/CorpusID:159042192> (ACM, 2019).
24. Hamilton, W. L., Ying, R. & Leskovec, J. Inductive representation learning on large graphs. In *Proc. 31st International Conference on Neural Information Processing Systems, NIPS'17*, 1025–1035 (Curran Associates Inc., 2017).
25. Tang, J. et al. Line: large-scale information network embedding. In *Proc. 24th International Conference on World Wide Web, WWW'15*, 1067–1077. <https://doi.org/10.1145/2736277.2741093> (International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 2015).
26. Park, J. B., Malthody, V. S., Qureshi, Z. & Hwu, W.-m. Accelerating sampling and aggregation operations in GNN frameworks with GPU-initiated direct storage accesses. *Proc. VLDB Endowment* **17**, 1227–1240. <https://doi.org/10.14778/3648160.3648166> (2024).
27. Peng, B., Ding, Y., Xia, Q. & Yang, Y. Recurrent neural networks integrate multiple graph operators for spatial time series prediction. *Appl. Intell.* **53**, 26067–26078 (2023).
28. Jin, M. et al. A survey on graph neural networks for time series: forecasting, classification, imputation, and anomaly detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **46**, 10466–10485 (2024).
29. Lai, X., Zhang, Z., Zhang, L., Lu, W. & Li, Z. Dynamic graph-based bilateral recurrent imputation network for multivariate time series. *Neural Netw.* **186**, 107298 (2025).
30. Liu, Q. et al. Gated spiking neural p systems for time series forecasting. *IEEE Trans. Neural Netw. Learn Syst.* **34**, 6227–6236 (2023).
31. Lin, K., Pan, J., Xi, Y., Wang, Z. & Jiang, J. Vibration anomaly detection of wind turbine based on temporal convolutional network and support vector data description. *Eng. Struct.* **306**, 117848 (2024).
32. Buu, S.-J. & Cho, S.-B. Time series forecasting with multi-headed attention-based deep learning for residential energy consumption. *Energies* **13**, 4722 (2020).
33. Ying, R. et al. Graph convolutional neural networks for web-scale recommender systems. In *Proc. 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD'18*, 974–983. <https://doi.org/10.1145/3219819.3219890> (Association for Computing Machinery, 2018)
34. Wang, J. et al. Metro passenger flow prediction via dynamic hypergraph convolution networks. *IEEE Trans. Intell. Transport. Syst.* **22**, 7891–7903 (2021).
35. Wang, P., Zhang, Y., Hu, T. & Zhang, T. Urban traffic flow prediction: a dynamic temporal graph network considering missing values. *Int. J. Geogr. Inf. Sci.* **37**, 885–912 (2023).
36. Vrahatis, A. G., Lazaros, K. & Kotsiantis, S. Graph attention networks: a comprehensive review of methods and applications. *Future Internet* **16** <https://www.mdpi.com/1999-5903/16/9/318> (2024).
37. Wang, X. et al. Traffic flow prediction via spatial temporal graph neural network. In *Proc. Web Conference 2020, WWW'20*, 1082–1092 (Association for Computing Machinery, 2020).
38. Wu, Z., Pan, S., Long, G., Jiang, J. & Zhang, C. Graph wavenet for deep spatial-temporal graph modeling. In *Proc. Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, 1907–1913. <https://doi.org/10.24963/ijcai.2019/264> (International Joint Conferences on Artificial Intelligence Organization, 2019).
39. Shao, Z. et al. A novel train-bridge interaction computational framework based on a meshless box girder model. *Adv. Eng. Softw.* **192**, 103628 (2024).
40. Zhao, H. et al. A velocity-related running safety assessment index in seismic design for railway bridge. *Mech. Syst. Signal Process* **198**, 110305 (2023).
41. Veličković, P. et al. Graph Attention Networks. <https://doi.org/10.48550/arXiv.1710.10903> (2017).
42. Vaswani, A. et al. Attention is all you need. In *Proc. 31st International Conference on Neural Information Processing Systems, NIPS'17*, 6000–6010 (Curran Associates Inc., 2017).
43. Peng, X. et al. Graph-based attention model for predictive analysis in train-bridge systems. *Appl Soft Comput* **179**, 113360 (2025).

44. Zhao, H. et al. Assessment of train running safety on railway bridges based on velocity-related indices under random near-fault ground motions. *Structures* **57**, 105244 (2023).
45. Zhao, H., Wei, B., Jiang, L. & Xiang, P. Seismic running safety assessment for stochastic vibration of train–bridge coupled system. *Arch. Civ. Mech. Eng.* **22**, 180 (2022).
46. Pang, R., Xu, B., Zou, D. & Kong, X. Stochastic seismic performance assessment of high CFRDs based on generalized probability density evolution method. *Comput. Geotech.* **97**, 233–245 (2018).
47. Harris, I., Osborn, T. J., Jones, P. & Lister, D. Version 4 of the CRU TS monthly high-resolution gridded multivariate climate dataset. *Sci. Data* **7**, 109 (2020).
48. Wang, X. Pems03 and pems04. <https://doi.org/10.21227/sf03-gz94> (2024).
49. Liu, S., Zhu, J., Lei, W. & Zhang, P. Spatial-temporal attention graph wavenet for traffic forecasting. In *Proc. 5th International Conference on Data-driven Optimization of Complex Systems (DOCS)*, 1–8 (IEEE, 2023).

Acknowledgements

The work described in this paper was encouraged by the National Natural Science Foundation of China (Grant no. 12572236), the Key R&D Projects of Hunan Province (no. 2024AQ2018), and the open fund of Chongqing Key Laboratory of Earthquake Resistance and Disaster Prevention of Engineering Structures of Chongqing University.

Author contributions

The code and paper were written by Xuan Peng and Ping Xiang. The model is built by Xuan Peng. The analysis and experiments of the paper were collaboratively completed by Peng Zhang, Zhuo Huang, Zefeng Liu, Yufei Chen, Zhanjun Shao, Han Zhao, Xiaonan Xie, Lizhong Jiang, and Zhuo Huang. The paper is reviewed by Lizhong Jiang, Zhouzhou Pan, Jianwei Yan, Binbin Yin, and Ping Xiang. The author team and project group are led by Ping Xiang.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at <https://doi.org/10.1038/s41467-025-64984-w>.

Correspondence and requests for materials should be addressed to Ping Xiang.

Peer review information *Nature Communications* thanks Charikleia D. Stoura, and the other anonymous, reviewer(s) for their contribution to the peer review of this work. A peer review file is available.

Reprints and permissions information is available at <http://www.nature.com/reprints>

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

© The Author(s) 2025