A Appendix

In this appendix, we provide a more detailed introduction to the datasets, describe the evaluation metrics for baselines, explain the LRP method in detail, and show additional experiments to demonstrate the superiority of the proposed method, DGExplainer.

A.1 Algorithm

Algorithm 1 DGExplainer

```
Input: The input $\{\mathbf{X}_t\}_{t=1}^T$ and $\{\mathbf{A}_t\}_{t=1}^T$, the final relevance $\{R_{\mathbf{h}_T^j}\}_{j=1}^N$, the pre-trained model $f(\cdot)$.
Output: The relevances $\{R_{\mathbf{X}_t}\}_{t=1}^T$
 1: // Forward process:
 2: for each $t \in [1, T]$ do
 3:     Compute $\hat{\mathbf{X}}_t$ via $\mathbf{F}_t^{(l+1)} = \sigma(\mathbf{V}_t\mathbf{F}_t^{(l)}\mathbf{W}_t^{(l)})$ with $\mathbf{F}_t^{(0)} = \mathbf{X}_t$, $\mathbf{F}_t^{(L)} = \hat{\mathbf{X}}_t$.
 4:     for each $j \in [1, N]$ do
 5:         Compute the hidden state $\mathbf{h}_t$ for the $j$-th sample
 6:         $\hat{\mathbf{X}}_t^{(j,:)}$ via Equation (2) with $\mathbf{h}_{t-1}$.
 7:     end for
 8: end for
 9: // Backward process:
10: for each $t = T, T-1, \dots, 1$ do
11:     for each $j \in [1, N]$ do
12:         Compute $R_{\mathbf{n}}, R_{\mathbf{n_1}}, R_{\mathbf{n_2}}$ via Equation (13), $R_{\mathbf{h}_{t-1}}$ via Equations (10) and (15), and $R_{\hat{\mathbf{X}}_t}$ for the $j$-th sample $\hat{\mathbf{X}}_t^{(j,:)}$ via Equation (14) and hence obtain $R_{\hat{\mathbf{x}}_t^j}$.
13:     end for
14:     Stack $\{R_{\hat{\mathbf{x}}_t^j}\}_{j=1}^N$ to get $R_{\hat{\mathbf{X}}_t}$.
15:     Calculate $R_{\mathbf{X}_t}$ by iteratively applying Equation (19) with $R_{\hat{\mathbf{X}}_t} = R_{\mathbf{F}_t^{(L)}}$.
16: end for
17: return $\{R_{\mathbf{X}_t}\}_{t=1}^T$.
```

A.2 Datasets

The statistics of the datasets and the initial performance of GCN-GRU on these datasets are summarized in Table 2.

Dataset	Reddit	PeMS04	PeMS08	Enron	FB	COLAB
# Nodes	55,863	307	170	184	663	315
# Edges	858,490	680	340	266	1068	308
# Train/Test	122/34	45/14	50/12	8/3	6/3	7/3
# Time Step	6	4	4	4	4	4
Performance	0.702	55.29	59.35	0.951	0.870	0.879

Table 2: Dataset statistics and performance metrics of the GCN-GRU model. We report the AUC (%) for the Reddit, Enron, FB, and COLAB datasets, and the MAE for the PeMS04 and PeMS08 datasets.

- **Reddit** is a directed network extracted from posts that generate hyperlinks connecting one subreddit to another. It includes various features, such as the source post, target URL, post title, and comment text, along with metadata like the number of upvotes and downvotes each post and comment received. The Reddit Hyperlink dataset comprises hyperlink information from over 3 million posts and their associated comments on the social media platform Reddit, spanning from 2008 to 2016.
- **PeMS04** and **PeMS08** are real-time traffic flow datasets providing traffic information for the state of California, USA. The PeMS04 dataset includes traffic flow data from over 39,000 sensors, while the PeMS08 dataset includes data from over 40,000 sensors. These sensors are located on freeways and arterial roads throughout California. The datasets cover the periods from January 1, 2018, to December 31, 2018, and from January 1, 2020, to December 31, 2020, respectively. Both datasets are collected at 5-minute intervals and include information on traffic speed, occupancy, and volume, resulting in 288 data points per detector per day. Additionally, the datasets include weather information and incident reports, which can be used to analyze the impact of weather and incidents on traffic flow. The data are transformed using zero-mean normalization to ensure the average is 0.
- **Enron**, **FB**, and **COLAB**: These datasets are dynamic graphs constructed from different types of interactions: email messages exchanged between employees, co-author relationships among authors, and Facebook wall posts, respectively. The Enron dataset represents the email communication network of employees at the Enron Corporation, where nodes represent individuals and edges represent email messages sent between them over time. The FB dataset captures the social network of Facebook users, where nodes represent users and edges represent friendship connections. Finally, the COLAB dataset contains transcripts of meetings held by community organizations, where nodes represent participants and edges represent their interactions during the meetings. We collected and processed these three datasets following the methodology described in [14].

A.3 Baselines

We provide detailed descriptions of the baselines used for comparison in our experiments in the follows:

- **Sensitivity Analysis (SA)** [2] computes importance scores using squared gradients of input features through back-propagation. It assumes that higher absolute gradient values indicate greater importance, but it fails to accurately represent importance and is prone to saturation issues [37].
- **GradCAM** [33] extends the Class Activation Mapping (CAM) [52] method to graph classification by removing the global average pooling layer constraint and mapping the final node embeddings to the input space for measuring node importance. It uses gradients as weights to combine different feature maps, computed by averaging the gradients of the target prediction with respect to the final node embeddings.
- **GNN-GI** [32] adopts Grad$\odot$Input (GI) [37], which quantifies the contribution of features by computing the element-wise product of the input features and the gradients of the decision function with respect to those features. As a result, GI takes into account both the sensitivity of features

and the scale of their values.

- **GNNExplainer** [46] generates explanations for predictions in the form of subgraphs and feature masks that highlight the relevant parts of the input data. It provides explanations by generating a compact subgraph from the input graph, along with a select subset of node features that greatly influence the prediction.
- **PGExplainer** [22] leverages a deep neural network parameterized explainer to generate global explanations that highlight important subgraphs influencing a model's predictions. This method endows PGExplainer with a natural capacity to deliver multi-instance explanations.
- **SubgraphX** [51] identifies important subgraphs measured by Shapley values. It employs the Monte Carlo tree search algorithm for efficiently exploring various subgraphs within a given input graph.
- **GCN-SE** [9] computes the importance of different graph snapshots by measuring the change in accuracy after masking the attention in that timestep.
- **T-GNNExplainer** [43] finds a subset of historical events that lead to the prediction, given a temporal prediction of a model. This method regards a temporal graph as a sequence of temporal events between nodes.

A.4 Implementation Details

We conducted all our experiments on a Linux machine equipped with four NVIDIA RTX A4000 Ti GPUs, each with 16GB of RAM. We used a two-layer GCN and trained the model for 1000 epochs using the Adam optimizer [17], with an initial learning rate of 0.01. We use the ReLU as the activation function for all layers, except for the output layer for the GCN-GRU model, where the softmax function is applied. For the link prediction task, we employed a two-layer MLP with 64 hidden units. We tested the stabilizer ϵ with values $\{1e\text{-}5, 1e\text{-}4, 1e\text{-}3, 1e\text{-}2, 1e\text{-}1, 1, 2\}$, and finally choose $\epsilon = 0.1$ to implement the proposed method. In stability experiments, we set r to $\{5\%, 10\%, 15\%, 20\%, 30\%\}$. The model performance results are based on the average analysis of 10 runs. The output embedding of a node u produced by the GCN-GRU model at time t is represented by $\mathbf{h}_t^u$.

A.5 Evaluation

Tasks. To comprehensively assess the effectiveness of our method, we conduct experiments on two representative dynamic graph tasks: link prediction and node regression.

- **Link Prediction**: For this task, we concatenate the feature embeddings of nodes u and v as $[(\mathbf{h}_T^u)^\top; (\mathbf{h}_T^v)^\top]^\top$ and use a multi-layer perceptron (MLP) to predict the link probability by optimizing the cross-entropy loss. We experiment with the Reddit, Enron, FB, and COLAB datasets and use the Area Under the Curve (AUC) as the evaluation metric. Following a previous study [25], in the implementation of the MLP, we use a rectified linear unit (ReLU) as the activation function for all layers except the output layer, where we apply the softmax function.
- **Node Regression**: To predict the value for a node u at time t, we apply a softmax activation function to the last layer of the GCN, resulting in the probability vector $\mathbf{h}_t^u$. We use the PeMS04 and PeMS08 datasets for this task and evaluate the performance using the mean absolute error (MAE) metric.

Metrics. We present a comprehensive overview of the four key quantitative metrics that have been instrumental in our analysis: confidence, sparsity, stability, and fidelity. The subsequent sections provide a detailed exposition of each metric.

- **Fidelity** characterizes whether the explanations are faithfully important to the model predictions [28]. In the experiment, we measure fidelity by calculating the difference in classification accuracy or regression errors obtained by occluding all nodes with importance values greater than a threshold τ_1 on a scale of $(0, 1)$. We averaged the fidelity across classes for each method. This approach is equivalent to the Fidelity+ metric proposed by Zheng *et al.* [56].
- **Surrogate Fidelity** [56] addresses the Out-of-Distribution (OOD) issue in resultant subgraphs caused by removing edges from the original graph. It introduces surrogate fidelity metrics to mitigate this problem. Specifically, for the previously defined fidelity+ metric, the surrogate version employs a function E_α that stochastically samples a graph by retaining each edge with probability α and removing it with probability $1 - \alpha$. Empirical studies demonstrate that selecting $\alpha < 1$ provides a more accurate fidelity measure, as it effectively reduces the OOD issue.
- **Sparsity** measures the fraction of nodes selected for an explanation [26, 51]. It evaluates whether the model efficiently marks the most contributive part of the dataset. High sparsity scores indicate that fewer nodes are identified as important. In our experiment, we compute sparsity by calculating the ratio of nodes with saliency values or relevances lower than a predefined threshold τ_2 on a scale of $(0, 1)$.
- **Stability** evaluates the consistency of explanations when small perturbations are applied to the input [28]. Explanations are considered stable if they remain approximately unchanged under such perturbations. To assess stability, we randomly add edges to the graph at a ratio of $r\%$ and measure the resulting changes in the relevances or importances produced by the model.

A.6 More Details About Layer-wise Relevance Propagation

Layer-wise relevance propagation (LRP) [1] is a technique for explaining the predictions of deep neural networks. It operates on the assumption that a neuron's relevance is proportional to its weighted activation value. This follows the intuition that a larger output activation indicates that the neuron carries more information and contributes more significantly to the final result.

Given an image x and a classifier $f(\cdot)$ the aim of layer-wise relevance propagation is to assign each pixel p of x a pixel-wise relevance $R_p^{(1)}$ such that:

$$f(x) \approx \sum_p R_p^{(1)}. \tag{20}$$

Pixels p with $R_p^{(1)} < 0$ contain evidence against the presence of a class, while $R_p^{(1)} > 0$ is considered as evidence for

		Evolve-GCN			DySAT			GC-LSTM			ROLAND		
		T-GNNE	DyExplainer	Ours	T-GNNE	DyExplainer	Ours	T-GNNE	DyExplainer	Ours	T-GNNE	DyExplainer	Ours
Reddit	Fidelity ↑	0.27	0.30	**0.32**	0.22	0.28	**0.31**	0.18	0.26	**0.27**	0.33	**0.43**	0.42
	Sparsity ↑	0.74	0.86	**0.88**	0.72	0.84	**0.85**	0.78	**0.91**	0.90	0.75	0.80	**0.81**
	Stability ↓	0.25	0.19	**0.14**	0.23	0.16	**0.15**	0.22	0.20	**0.18**	0.27	0.25	**0.21**
PeMS04	Fidelity ↑	0.35	0.41	**0.43**	0.15	**0.23**	0.22	0.33	0.35	**0.36**	0.27	**0.30**	**0.30**
	Sparsity ↑	**0.99**	0.95	**0.99**	0.96	0.92	**0.99**	0.94	0.95	**0.99**	0.97	**0.99**	**0.99**
	Stability ↓	0.14	0.16	**0.11**	0.33	0.35	**0.30**	0.30	0.29	**0.27**	**0.19**	0.25	0.23
PeMS08	Fidelity ↑	0.21	0.26	**0.31**	0.15	0.20	**0.22**	0.16	**0.21**	**0.21**	0.28	0.31	**0.32**
	Sparsity ↑	**0.98**	0.94	0.95	0.87	0.86	**0.90**	0.88	0.89	**0.91**	0.85	**0.91**	0.90
	Stability ↓	0.18	0.22	**0.16**	0.21	0.20	**0.17**	0.19	0.20	**0.15**	0.18	0.14	**0.11**
Enron	Fidelity ↑	**0.21**	0.23	0.24	**0.17**	0.19	0.20	**0.22**	0.25	0.24	**0.24**	0.25	0.27
	Sparsity ↑	0.78	0.83	**0.87**	0.80	0.79	**0.83**	0.76	0.81	**0.82**	0.70	0.76	**0.79**
	Stability ↓	0.24	0.22	**0.16**	0.22	0.19	**0.11**	0.20	0.23	**0.19**	0.11	0.15	**0.08**
FB	Fidelity ↑	0.23	0.27	**0.33**	0.33	0.34	**0.37**	0.19	0.21	**0.23**	0.09	**0.13**	0.11
	Sparsity ↑	0.89	0.95	**0.96**	0.77	0.82	**0.87**	0.86	0.91	**0.94**	0.88	0.89	**0.92**
	Stability ↓	0.15	0.13	**0.11**	0.21	0.19	**0.17**	0.25	0.23	**0.19**	0.19	**0.14**	0.15
COLAB	Fidelity ↑	0.39	0.43	**0.47**	0.38	**0.44**	0.43	0.37	0.39	**0.41**	0.32	0.36	**0.38**
	Sparsity ↑	0.94	0.97	**0.98**	0.92	0.94	**0.97**	0.91	**0.99**	0.98	0.92	0.97	**0.99**
	Stability ↓	0.19	0.15	**0.11**	0.21	0.19	**0.17**	0.22	**0.19**	**0.19**	**0.14**	0.16	0.15

Table 3: Experimental results on other dynamic GNNs, in terms of fidelity ($\tau_1 = 0.8$), sparsity ($\tau_2 = 3 \times 10^{-4}$), and stability ($r = 20\%$). For each metric, the best results are highlighted in **bold** text.

the presence of a class. These pixel-wise relevances can be visualized as an image called a heatmap. Obviously, many possible decompositions exist that satisfy Equation (20). One work yields pixel-wise decompositions that are consistent with evaluation measures and human intuition.

The objective described in Equation (20) can be easily extended to tasks beyond image classification. For instance, in this paper, we study the node classification task where $f(\cdot)$ represents a GNN or dynamic GNN. Here, the goal is to compute the relevance of each feature p for every node x as specified in Equation (20). Further details on applying Equation (20) to other tasks are provided in Appendix A.6.

Epsilon Rule (LRP-ϵ) [1] A first enhancement of the basic LRP-0 rule consists of adding a small positive term ϵ in the denominator: The work in [1] established two formulas for computing the messages $R_{k_1 \leftarrow k_2}^{(l,l+1)}$. The first formula called ϵ-rule is given by

$$R_{k_1 \leftarrow k_2}^{(l,l+1)} = \frac{z_{k_1 k_2}}{z_{k_2} + \epsilon \cdot \operatorname{sign}(z_{k_2})} R_{k_2}^{(l+1)}, \tag{21}$$

where $z_{ij} = (w_{ij} x_i)^p$ and $z_j = \sum_{k: w_{kj} \neq 0} z_{kj}$. Here, $R_{k_2}^{(l+1)}$ denotes the relevance of neuron k_2 at layer $l+1$, and $R_{k_1 \leftarrow k_2}^{(l,l+1)}$ represents the portion of relevance attributed to neuron k_1 at layer l by neuron k_2. The activation of neuron k_1 at layer l is denoted by $a_{k_1}^{(l)}$. The stabilizing term ϵ is introduced to avoid division by zero in the denominator.

The concept behind LRP assumes that the relevance, denoted as $R_{k_2}^{(l+1)}$, is known for a neuron in the subsequent layer $(l+1)$. This assumption allows us to break down and distribute this relevance to the neurons, denoted as k_1, in the current layer l that contribute input to the neuron k_2. This process enables us to determine the relevance value for a neuron k_1 in layer l by aggregating all the incoming messages from neurons in layer $(l+1)$. A notable challenge in LRP is formulating an appropriate rule for redistributing relevance across each layer. The role of ϵ is to absorb some relevance when the contributions to the activation of neuron k are weak or contradictory. As ϵ becomes larger, only the most salient explanation factors survive the absorption. This typically leads to explanations that are sparser in terms of input features and less noisy.

By summing up the relevance over all neurons k_2 in layer $(l+1)$, based on Equation (21). The final propagation rule can be obtained from Equation (21):

$$R_{k_1 \leftarrow k_2}^{(l,l+1)} = \sum_{k_2} \frac{\mathbf{W}_{k_1 k_2} a_{k_1}^{(l)}}{\epsilon + \sum_{k_1} \mathbf{W}_{k_1 k_2} a_{k_1}^{(l)}} R_{k_2}^{(l+1)}, \tag{22}$$

where $\mathbf{W}_{k_1 k_2}$ represents the connection weight between neurons k_1 and k_2.

Clearly, the connection between the relevance and the weighted activation $\mathbf{W}_{k_1 k_2} a_{k_1}^{(l)}$ is discernible. This relationship indicates that the relevance varies in proportion to the magnitude of the weighted activation. Additionally, the nature of the contribution, whether positive or negative, depends

Method	Metric	Reddit	PeMS04	PeMS08	Enron	FB	COLAB
`DGExplainer`	Fidelity ↑	0.42	0.39	0.30	0.23	0.36	0.53
	Sparsity ↑	0.87	0.99	0.95	0.85	0.96	0.96
	Stability ↓	0.13	0.15	0.12	0.15	0.12	0.18
LRP	Fidelity ↑	0.29	0.27	0.25	0.20	0.30	0.48
	Sparsity ↑	0.72	0.95	0.93	0.80	0.95	0.94
	Stability ↓	0.18	0.21	0.16	0.12	0.17	0.19

Table 4: An ablation study comparing the proposed `DGExplainer` with the direct application of LRP, which does not account for the temporal information in the GRU module. This experiment evaluates fidelity ($\tau_1 = 0.8$), sparsity ($\tau_2 = 3 \times 10^{-4}$), and stability ($r = 20\%$).

on the sign of the weighted activation. In our implementation, we use the softmax activation function, ensuring that the weighted activations are in the range $(0, 1)$. As a result, the denominator will always be non-zero after adding ϵ.

A.7 More Experiments on Dynamic GNN Architectures

We conducted additional experiments on diverse dynamic GNN architectures, including, Evolve-GCN [25], DySAT [29], GC-LSTM [5], and ROLAND [47]. From the results in Table 3, we can observe that Evolve-GCN generally offers higher fidelity and is easier to explain compared to other architectures. In contrast, other dynamic GNNs, such as ROLAND, show varying levels of explainability difficulty; for instance, ROLAND is particularly challenging to explain on the FB dataset. Overall, while other methods may produce more accurate and stable explanations, they do so with higher fidelity and sparsity, but at the cost of lower stability.

A.8 Ablation Study

To demonstrate the importance of capturing temporal dependencies in explaining dynamic graphs, we perform an ablation study by treating the hidden representation H_t as the output of the GNN, denoted by $\hat{\mathbf{X}}_{t-1}$, without backpropagating the relevance through the GRU layer. As illustrated in Figure 3, Stage 1 is omitted, and the relevance of the hidden representation is directly used as the input for Stage 2. We apply LRP only to the GNN modules at each time step to obtain the relevance of the input features, resulting in $R_{\hat{\mathbf{X}}_t} = R_{\hat{\mathbf{X}}_{t-1}}$. This approach does not account for the relevance back-propagation process within the GRU module, which removes the temporal information needed to explain the dynamic GNNs, specifically in the GCN-GRU model.

From the results in Table 4, we observe that, generally, LRP exhibits lower fidelity and sparsity compared to `DGExplainer`. This suggests that `DGExplainer`, which back-propagates relevance through the GRU, more accurately identifies important features for predictions. Additionally, the stability of `DGExplainer` is comparable to that of directly applying LRP on the COLAB dataset, and is smaller than many of the baselines in Table 1. This experiment validates the effectiveness of utilizing time-dependent information, which reveals clearer relevance patterns compared to treating each graph independently.

A.9 Additional Baseline

We conducted additional experiments using GNN-LRP [32], and the results are below:

Method	Metric	Reddit	PeMS04	COLAB
GNN-LRP	Fidelity ↑	0.37	0.35	0.49
	Fidelity+ ↑	0.25	0.27	0.31
	Sparsity ↑	0.82	0.99	0.74
	Stability ↓	0.15	0.19	0.20

Table 5: Additional experiments on GNN-LRP evaluated for fidelity ($\tau_1 = 0.8$), fidelity+ ($\tau_1 = 0.8$), sparsity ($\tau_2 = 3 \times 10^{-4}$), and stability ($r = 20\%$).

From Table 5, we observe that 1) GNN-LRP exhibits lower fidelity and sparsity compared to `DGExplainer`. 2) The stability of `DGExplainer` is higher than that of GNN-LRP, indicating that GNN-LRP is less stable.

A.10 Case Study

To demonstrate that the proposed `DGExplainer` can capture time-varying dependencies among snapshots, we visualize the relevance of several nodes across different time steps using the Reddit dataset.

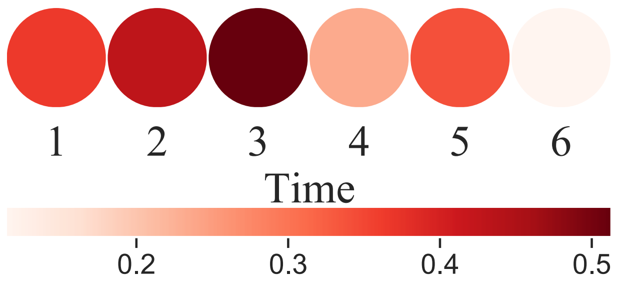

Figure 7: Heatmap of the node #67 over six time steps in the Reddit dataset.

From Figure 6, we observe that nodes #43, #67, #80, and #91 maintain relatively consistent importance across time steps 1 to 3, highlighting a continuity in node importance over time within dynamic graphs. Furthermore, node #91 shows lower importance in the earlier snapshots (1 to 3), but its significance increases at time step 6, suggesting that node impor-

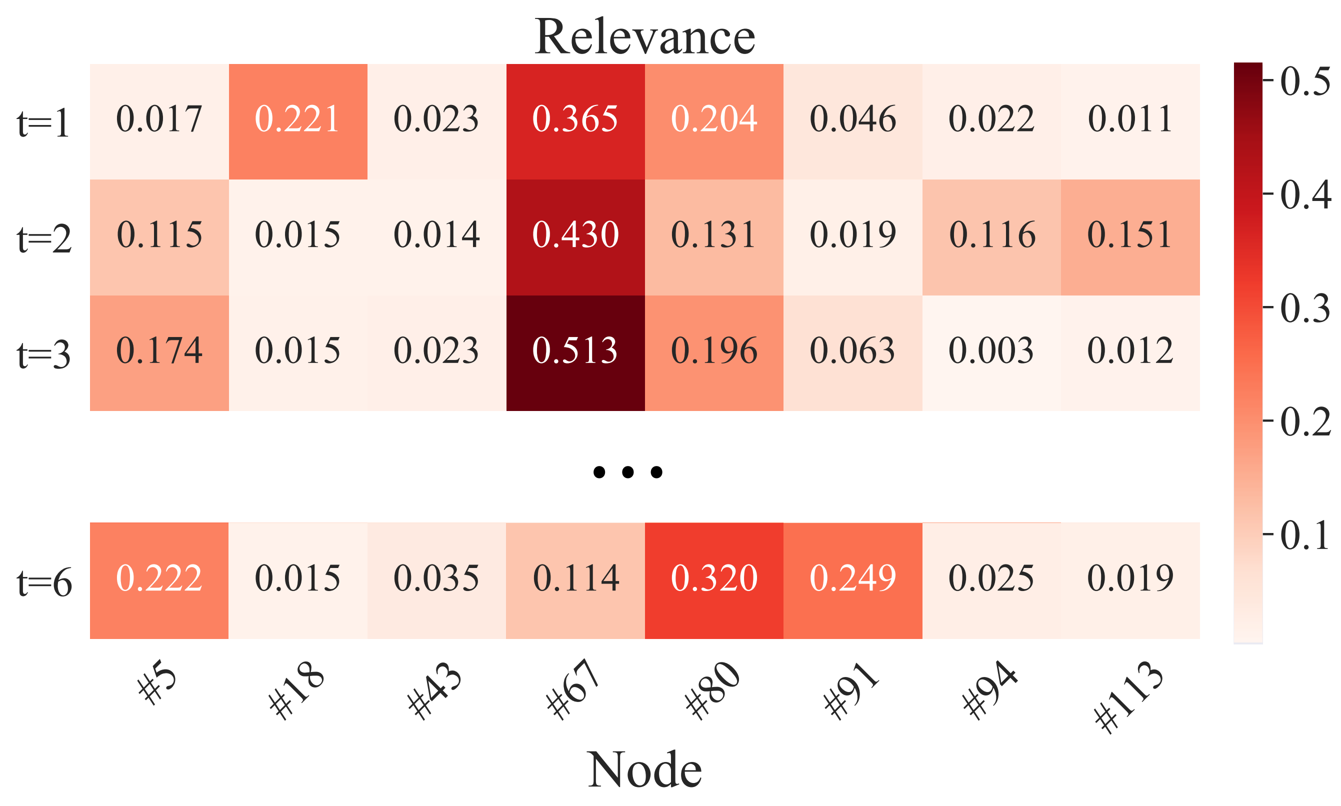

Figure 6: Heatmap showing the relevance of eight nodes from time step 1 to 6 on the Reddit dataset. Darker shades represent higher node relevance at each time step.

tance can change over time. To explore the evolution of relevance further, we also visualize the temporal relevance of a specific node, #67, over six consecutive time steps, as shown in Figure 7. We observe that node #67 reaches its peak relevance at time step 3. Its relevance gradually increases before time step 3, begins to decrease at time step 4, and rebounds slightly at time step 5.

A.11 Complexity analysis

We will include a complexity analysis, covering both runtime and memory usage in this subsection.

Runtime. To assess the computational efficiency of our method, we report its runtime on six datasets used in our experiments. The reported runtime corresponds to the total time required to generate explanations for all test samples using layer-wise back-propagation. This evaluation provides insight into the scalability of our method across datasets of varying sizes and temporal resolutions.

Dataset	Reddit	PeMS04	PeMS08	Enron	FB	COLAB
Runtime (s)	152.4	18.7	12.3	9.8	27.5	21.9

Table 6: Runtime (in seconds) of our method using layer-wise back-propagation to explain dynamic GNNs on each dataset.

As shown in Table 6, the runtime of our method scales with the size and complexity of the dataset. While larger datasets such as Reddit incur higher computational costs, smaller datasets like Enron and PeMS08 require significantly less runtime.

Memory Usage. To evaluate the memory efficiency of our method, we report the peak GPU memory usage during explanation generation on all six datasets. This analysis helps illustrate the hardware requirements of our approach and its scalability to larger graphs and longer temporal sequences.

Dataset	Reddit	PeMS04	PeMS08	Enron	FB	COLAB
Usage (GB)	11.2	4.3	3.9	3.1	5.7	5.2

Table 7: Peak GPU memory usage (in GB) of our method on each dataset during explanation generation.

As shown in Table 7, our method exhibits moderate memory usage across all datasets, with peak consumption remaining well within the 16GB GPU limit. While larger datasets like Reddit naturally demand more memory, the method remains efficient and feasible for real-world applications across datasets of varying scales.

A.12 More Hyperparameter Study

To evaluate the sensitivity of our method to the stabilizing term ϵ in the relevance propagation rule, we conduct an ablation study by varying ϵ across a range of values. The stabilizer ϵ plays a crucial role in avoiding division by zero and controlling the numerical stability of the explanation scores. However, its value may also influence the trade-offs between fidelity, sparsity, and stability. We examine the effect of $\epsilon \in \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 2\}$ on the performance of our method using the same evaluation setup as in the main experiment (shown in Table 1).

ϵ	Fidelity ↑	Fidelity+ ↑	Sparsity ↑	Stability ↓
10^{-5}	0.29	0.16	0.81	0.20
10^{-4}	0.31	0.18	0.81	0.19
10^{-3}	0.35	0.21	0.83	0.17
10^{-2}	0.38	0.23	0.84	0.15
10^{-1}	0.40	0.26	0.87	0.14
1	0.41	0.25	0.86	0.14
2	0.40	0.25	0.86	0.14

Table 8: Effect of varying the stabilizing term ϵ on the performance of our method on Reddit dataset.

As shown in Table 8, the choice of the stabilizing term ϵ significantly influences the performance of our method across all four metrics. When ϵ is too small (e.g., 10^{-5} or 10^{-4}), both fidelity and fidelity+ drop noticeably, and stability deteriorates due to numerical instability in the relevance propagation process. As ϵ increases, these metrics improve, with performance peaking around $\epsilon = 0.1$, which achieves a favorable balance: high fidelity (0.403), high sparsity (0.862), and low stability score (0.135). Beyond this point, increasing ϵ further to 1 or 2 leads to a slight drop in fidelity and sparsity, accompanied by a mild increase in instability. These results suggest that moderate values of ϵ (e.g., 10^{-2} to 10^{-1}) are optimal for ensuring stable and effective relevance propagation.

A.13 Related Work

We review previous studies related to our work, focusing first on the recent advances in dynamic graph neural networks and then on existing explainability methods for static GNNs.

Dynamic Graph Neural Networks

Dynamic graph neural networks (Dynamic GNNs) consider both temporal and graph-structural information to tackle dynamic graphs. These networks are commonly applied in social media, citation networks, transportation networks, and pandemic networks. DANE [21] is an efficient dynamic GNN that updates node embeddings using the eigenvectors of the graph's Laplacian matrix, based on the graph from the previous time step. CTDANE [24] and NetWalk [49] extend random walk-based approaches by enforcing temporal rules on the walks. Additionally, embedding methods aggregate neighboring node features. For example, DynGEM [10] and Dyngraph2vec [11] use deep autoencoders to encode snapshots of dynamic graphs.

A prevalent category of approaches combines GNNs with recurrent architectures, where the GNNs extract graph-structural information and the recurrent units handle sequential flows. GCRN [34] leverages GCN layers to obtain node embeddings and feeds them into recurrent layers to track dynamism. STGCN [48], which stacks ST-Conv blocks, proposes a sophisticated architecture that effectively captures complex localized spatial-temporal correlations. Instead of directly integrating RNNs into the entire structure, EvolveGCN [25] uses RNNs to update the weights of GCNs. Another approach [14] introduces variational autoencoder versions for dynamic graphs, VGRNN and SI-VGRNN. Both models use a GCN integrated into an RNN as an encoder to track the temporal evolution of the graph. The GCN-GRU model used to demonstrate the proposed method has wide applications [12, 45, 54]. The two modules are co-trained to capture the spatial-temporal information in dynamic graphs. For example, in traffic flow prediction, the GCN models the dynamics of traffic as an information dissemination process. Meanwhile, the GRU captures dependencies across different time steps through gate units that are trained to manage inputs and memory states, enabling the retention of information over longer periods [54].

Explainability on GNNs

Existing explainability approaches for GNNs are mostly focused on static GNNs and can be categorized into four main directions. The first direction focuses on highlighting the importance of various input features by analyzing feature gradients and relevances. For instance, Sensitivity Analysis [2] assigns importance scores to different input features using the squared values of their gradients. GNN-GI employs the Grad$\odot$Input method [28, 36], which calculates feature contribution scores as the element-wise product of the input features. GNN-LRP [32] assigns a relevance score to each walk, representing a message flow path within the graph. This relevance score is determined using the T-order Taylor expansion of the model with respect to the incorporation operator. A recent study [44] introduces two novel relevant walk search algorithms based on max-product message passing, reducing the computational complexity of GNN-LRP from exponential to polynomial time. Additionally, GradCAM [26] adapts the Class Activation Mapping (CAM) approach [26] for general graph classification models by removing the requirement for a global average pooling layer.

The second research direction focuses on accumulating local effects to create a locally faithful approximation. For example, GraphLime [15] utilizes neighboring nodes as perturbed inputs and employs a nonlinear surrogate model. This model is capable of assigning significant weights to features that are crucial for inference. In addition, PGM-Explainer [41] leverages an interpretable Bayesian network to approximate the predictions needed for explanation.

The third line of research examines perturbation-based interpretation methods, which identify key components affecting model predictions by perturbing nodes, edges, or features. Specifically, GNNExplainer [46] and PGExplainer [22] identify key features by maximizing mutual information between the perturbed and input graphs. Graphsvx [8] and SubgraphX [51] use the Shapley value [35] to assess the importance of features and nodes in Graphsvx and subgraphs in SubgraphX. TempME [4] samples temporal motifs around the interaction between two nodes' predictions, assigning importance scores while balancing explanation accuracy and information compression by maximizing mutual information with the target prediction and minimizing it with the original temporal graph. SubMT [6] introduces an interpretable subgraph learning method that identifies the subgraph multilinear extension as a factorized distribution. The extracted subgraph is considered interpretable and faithful when its prediction strongly correlates with the subgraph's sampling probability. GraphMask [30] trains a classifier to generate edge masks for each GNN layer, identifying edges that can be removed without altering the model's predictions.

The final direction focuses on model-level explanations, offering a broader understanding of how the model operates as a whole. This method provides general insights into the overall functioning of the model, rather than focusing on individual predictions. For example, XGNN [50] offers a global explanation of GNNs by training a graph generator to create patterns that maximize the predictions of a given model.

Despite the success of existing explainability methods, they primarily focus on static GNNs and cannot be directly applied to explain dynamic GNNs because they overlook the temporal or dynamic aspects of graphs. Recently a dynamic GNN explainer, DyExplainer [42], has been proposed. Specifically, DyExplainer trains a dynamic GNN to produce node embeddings and uses structural and temporal attention by capturing both pivotal relationships within snapshots and temporal dependencies across long-term snapshots. While explaining dynamic GNNs remains a critical challenge, only a limited number of approaches have been proposed to address this issue. This motivates us to fill this gap by proposing our method.