Self-Explainable Graph Transformer for Link Sign Prediction

Anonymous submission

Abstract

Signed Graph Neural Networks (SGNNs) have been shown to be effective in analyzing complex patterns in real-world situations where positive and negative links coexist. However, SGNN models suffer from poor explainability, which limit their adoptions in critical scenarios that require understanding the rationale behind predictions. To the best of our knowledge, there is currently no research work on the explainability of the SGNN models. Our goal is to address the explainability of decision-making for the downstream task of link sign prediction specific to signed graph neural networks. Since post-hoc explanations are not derived directly from the models, they may be biased and misrepresent the true explanations. Therefore, in this paper we introduce a Self-Explainable Signed Graph transformer (SE-SGformer) framework, which can not only outputs explainable information while ensuring high prediction accuracy. Specifically, We propose a new Transformer architecture for signed graphs and theoretically demonstrate that using positional encoding based on signed random walks has greater expressive power than current SGNN methods and other positional encoding graph Transformer-based approaches. We constructs a novel explainable decision process by discovering the K-nearest (farthest) positive (negative) neighbors of a node to replace the neural network-based decoder for predicting edge signs. These K positive (negative) neighbors represent crucial information about the formation of positive (negative) edges between nodes and thus can serve as important explanatory information in the decision-making process. We conducted experiments on several real-world datasets to validate the effectiveness of SE-SGformer, which outperforms the state-of-the-art methods by improving 2.2% prediction accuracy and 73.1% explainablity accuracy in the best-case scenario. The code is provided in supplementary material.

Introduction

Signed Graph Neural Networks (SGNNs) are widely used for learning representations of signed graphs, as shown in Figure 1. Despite recent years have witnessed a growing interest in SGNNs with link sign prediction as the focal task (Derr, Ma, and Tang 2018; Shu et al. 2021; Zhang et al. 2023b; Ni et al. 2024), no existing study on SGNNs has addressed the issue of explainability, which hinders their adoption in crucial domains. For example, in financial networks, being able to interpret why certain transactions are flagged as suspicious can improve fraud detection and prevention. Therefore, understanding why certain relationships are classified as positive or negative can help SGNNs be more widely applied.

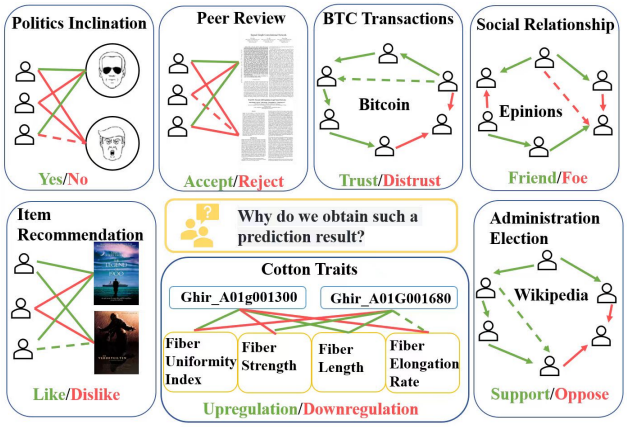
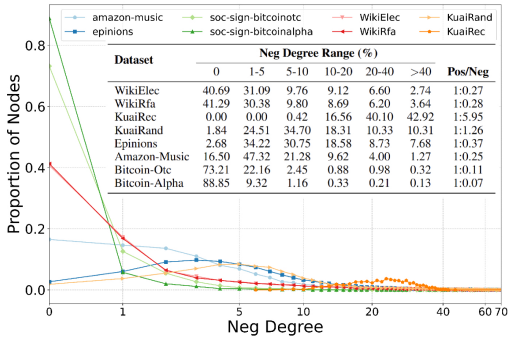


Figure 1: An illustration of signed graphs in real world. The dashed lines represent the edges to be predicted, and black and red lines represent positive and negative edges, resp.

Research on GNN explainability falls into two main categories: post-hoc explanations (Zhang et al. 2024; Yuan et al. 2022) and self-explainable approaches (Deng and Shen 2024; Seo, Kim, and Park 2024). Post-hoc explanation methods like GNNExplainer (Ying et al. 2019) and PGExplainer (Luo et al. 2020) offer interpretations for trained GNN models, but these explanations may be biased and not truly reflective of the model. As a result, current research is shifting toward self-explaining methods, where models generate explanations alongside predictions. For example, SE-GNN (Dai and Wang 2021) uses K-nearest labeled nodes for explainable node classification, while ProtGNN (Zhang et al. 2022) integrates prototype learning to enhance interpretability. While effective, these methods are primarily designed for unsigned graphs with graph or node classification tasks, making them unsuitable for SGNNs focusing on link sign prediction. This highlights the need for a new explainable framework for signed graph representation learning that can offer both predictions and explanations.

One potential approach for obtaining self-explanations in link sign prediction is to identify explainable K-nearest pos-



Dataset	Neg Degree Range (%)						
	0	1-5	5-10	10-20	20-40	>40	Pos/Neg
WikiElec	40.69	31.09	9.76	9.12	6.60	2.74	1:0.27
WikiRfa	41.29	30.38	9.80	8.69	6.20	3.64	1:0.28
KuaiRec	0.00	0.00	0.42	16.56	40.10	42.92	1:5.95
KuaiRand	1.84	24.51	34.70	18.31	10.33	10.31	1:1.26
Epinions	2.68	34.22	30.75	18.58	8.73	7.68	1:0.37
Amazon-Music	16.50	47.32	21.28	9.62	4.00	1.27	1:0.25
Bitcoin-Otc	73.21	22.16	2.45	0.88	0.98	0.32	1:0.11
Bitcoin-Alpha	88.85	9.32	1.16	0.33	0.21	0.13	1:0.07

Figure 2: Proportion of node negative degrees and Pos/Neg Ratios across datasets (%)

itive neighbors (and K-farthest negative neighbors) for each node. If the closeness between the two nodes of an edge e_{ij} is similar to the closeness between node v_i and its K-nearest positive (or K-farthest negative) neighbors, the predicted sign of the edge is positive (or negative). The identified K-nearest positive (or K-farthest negative) neighbors then serve as the explanatory information. This approach simultaneously provides a prediction result and an explanation for that prediction.

The key to identifying the K-nearest (farthest) positive (negative) neighbors for each node lies in learning proper node representations and selecting an adequate number of positive and negative neighbors for any given node. The challenges can be divided into the two aspects:

Challenge 1 (Encoding part): How to improve the quality of node representations. To prevent overfitting, current SGNN models limit networks to three layers or fewer, restricting their ability to capture multi-hop information (Derr, Ma, and Tang 2018). Meanwhile, GCN-based SGNN frameworks cannot learn proper representations from unbalanced cycles (Zhang et al. 2023b).

Challenge 2 (Explanation part): How to find a sufficient number of negative neighbors for nodes. As is shown in Figure 2, a significant proportion of nodes have few or even no negative neighbors. For example, in the Bitcoin-Alpha dataset, over 80% of the nodes have no negative neighbors.

For Challenge 1, We design a novel graph Transformer architecture specifically for signed graphs (see **Encoding Module** section). Specifically, We use signed random walk encoding to help capture multi-hop neighbor information. We theoretically demonstrate that this encoding method has a stronger representation power compared to current SGNNs and other common encoding methods, such as the shortest path encoding.

For Challenge 2, We employ signed diffusion matrix based on Signed Random Walk with Restart (SRWR) algorithm (Jung et al. 2016) (see **Explainable Prediction Module** section) to uncover potential negative relationships among nodes.

Overall, we propose a novel Self-Explainable Signed Graph Transformer (SE-SGformer) framework which utilizes the Transformer architecture as an encoder for the signed graph, predicting edge signs by identifying the K-nearest (farthest) positive (negative) neighbors and providing corresponding explanatory information. We conduct experiments on eight real-world datasets to validate the effectiveness of SE-SGformer, which surpasses state-of-the-art methods by achieving a 2.2% increase in prediction accuracy and a 73.1% improvement in explainability accuracy under optimal conditions. Related work can be found in Appendix. Our contributions are summarized as follows:

- We first proposed a self-explainable model SE-SGformer for signed graphs, which is specifically designed for the link sign prediction task and can provide explanations along with the predictions.
- We propose a novel graph Transformer-based representation learning model for signed graphs and theoretically prove that our random walk encoding is more powerful than the shortest path encoding and our Transformer architecture with random walk encoding is more powerful than SGCN.
- We conduct extensive experiments on several real-world datasets and prove that SE-SGformer achieve performance comparable to state-of-the-art models and achieve good explanatory accuracy.

Problem Definition

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}^+, \mathcal{E}^-)$ be a signed network, where $\mathcal{V} = \{v_1, v_2, \dots v_{|\mathcal{V}|}\}$ represents the set of $|\mathcal{V}|$ nodes while $\mathcal{E}^+ \subset \mathcal{V} \times \mathcal{V}$ and $\mathcal{E}^- \subset \mathcal{V} \times \mathcal{V}$ denote the sets of positive and negative links, respectively. For each edge $e_{ij} \in \mathcal{E}^+ \cup \mathcal{E}^-$ connecting two nodes v_i and v_j, the edge can be either positive or negative, but not both, implying $\mathcal{E}^+ \cup \mathcal{E}^- = \emptyset$. The graph structure can be described by an adjacency matrix $A \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$, where $A_{ij} = 1$ means there exists a positive link from v_i to v_j , $A_{ij} = -1$ denotes a negative link, and $A_{ij} = 0$ otherwise (meaning no link from v_i to v_j). Note that, real-world signed graph datasets typically do not provide node features. Therefore, there is no feature vector x_i associated with each node v_i.

The goal of an SGNN is to learn an *embedding function* $f_\theta : \mathcal{V} \to \mathcal{Z}$, which maps the nodes of a signed graph to a latent vector space $\mathcal{Z}$. In this space, $f_\theta(v_i)$ and $f_\theta(v_j)$ are close if $e_{ij} \in \mathcal{E}^+$ and distant if $e_{ij} \in \mathcal{E}^-$. Additionally, we adopt *link sign prediction* as the downstream task for SGNN, following mainstream studies. This task aims to infer the sign of a link given the nodes v_i and v_j. The link sign prediction can be explained as follows: (1) for a node pair (v_i, v_j) connected by a directed positive edge, taking node v_i as an example, v_i's K-nearest positive neighbors have a higher similarity score with respect to v_j; (2) for a node pair (v_i, v_j) connected by a directed negative edge, taking node v_i as an example, v_i's K-farthest negative neighbors have a higher similarity score with respect to v_j. With the aforementioned notations, we formualte our Self-explainable link sign prediction problem as:

Given a signed graph $\mathcal{G}$, the task is to learn the SGNN parameters θ while simultaneously *predicting link signs* and *generating explanations* by identifying the set of K-nearest (farthest) positive (negative) neighbors.

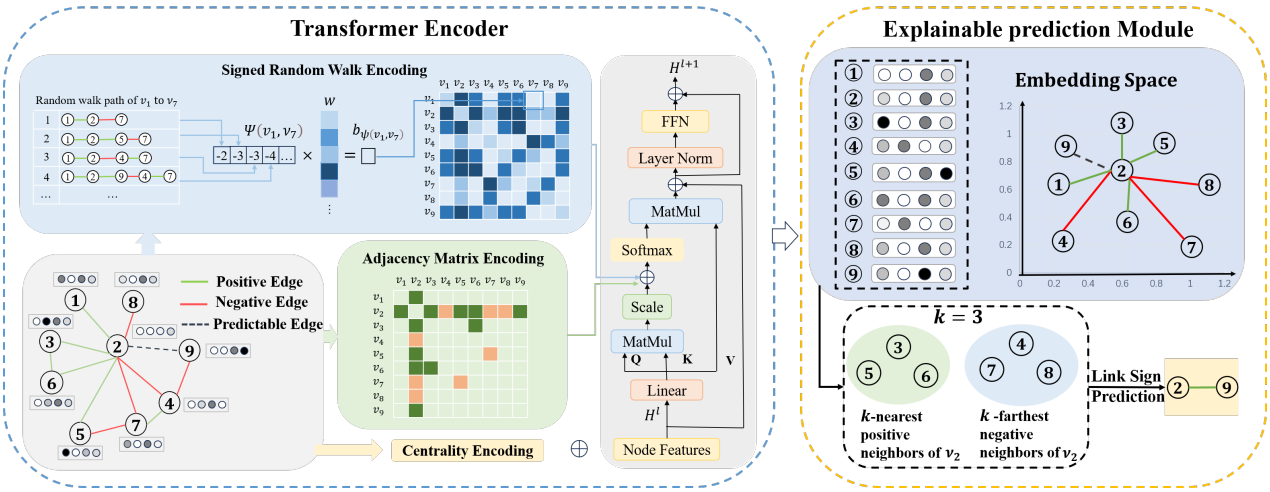


Figure 3: The overall architecture of SE-SGformer. Firstly, the Transformer encoder, equipped with centrality encoding, adjacency matrix encoding, and random walk encoding, obtain node embeddings. Next, in the explainable prediction module, the K-nearest positive (and K-farthest negative) neighbors of the nodes are identified and used to predict the unknown link sign.

Proposed Method

In this section, we introduce the details of the proposed framework SE-SGformer. The basic idea of SE-SGformer is to first use a Transformer to encode the signed graph. Then, for an edge with an unknown sign from node v_i to node v_j, we identify the K-nearest (farthest) positive (negative) neighbors of v_i. The sign of the edge e_{ij} is determined based on the similarity between node v_j and the positive or negative neighbors of node v_i, with explanatory information provided simultaneously. The overview of our proposed SE-SGformer is shown in Figure 3. It is mainly divided into two parts: encoding module and explainable prediction module.

Encoding Module

The classical Transformer architecture (Vaswani et al. 2017) is composed of self-attention modules and feed-forward neural networks. In the self-attention module, the input $H \in \mathbb{R}^{n\times d}$ are projected to the corresponding query Q, key K, and value V. The self-attention is then calculated as:

$$Q = HW_Q, \quad K = HW_K, \quad V = HW_V, \tag{1}$$

$$\tilde{A} = \frac{QK^T}{\sqrt{d_K}}, \mathrm{Attn}(H) = \mathrm{softmax}(\tilde{A})V \tag{2}$$

where $\tilde{A}$ denotes the matrix capturing the similarity between queries and keys. $W_Q \in \mathbb{R}^{d\times d_K}, W_K \in \mathbb{R}^{d\times d_K}, W_V \in \mathbb{R}^{d\times d_V}$ denote the projected matrices of query, key and value, respectively.

For graph representation, incorporating structural information of graphs into Transformer models is crucial. Therefore, We introduce three new encodings. The specific details are as follows:

Centrality Encoding. Degree centrality measures the influence of a node in a graph, where higher degrees indicate potentially greater influence. However, these insights are often overlooked in attention computations. Therefore, we have designed a centrality encoding tailored for signed graphs to reflect the importance of each node. To be specific, we assign two real-valued embedding vectors to each node based on its positive degree and negative degree. We directly add the centrality encoding of each node to its original features x_i to form the new node embeddings:

$$h_i^{(0)} = x_i + c^-_{\deg^-(v_i)} + c^+_{\deg^+(v_i)}, \tag{3}$$

where $c^-, c^+ \in \mathbb{R}^d$ are learnable embedding vectors determined by the negative degree $\deg^-(v_i)$ and positive degree $\deg^+(v_i)$, respectively.

Adjacency Matrix Encoding. Unlike the Transformer's approach to sequential data, where specific positional encodings are added to each word to indicate its position in a sentence, nodes in a graph are not arranged in a sequence. To encode the structural information of a graph in the model, we introduce the adjacency matrix encoding. A is the adjacency matrix of the graph, characterizing their direct positive and negative neighbors. After normalizing the adjacency matrix, we obtain the adjacency matrix encoding $\hat{A}$ which serves as a bias term in the self-attention module. Denote $\tilde{A}_{ij}$ as the (i, j)-element of the Query-Key product matrix $\tilde{A}$, we have:

$$\tilde{A}_{ij} = \frac{(h_iW_Q)(h_jW_K)^T}{\sqrt{d}} + \hat{A}_{ij}, \tag{4}$$

Signed Random Walk Encoding. Adjacency matrix encoding can only capture the relationships between nodes and their one-hop neighbors, but it cannot represent the relationships between nodes and their multi-hop neighbors. Previous works introduce spatial encoding to capture the relationships between a node and its multi-hop neighbors like the shortest path encoding (Ying et al. 2021). However, the shortest path encoding approach only considers a single path, without taking alternative routes into account. Therefore, we introduce the *signed random walk encoding* to exploit the relative position between nodes and their high-order neighbors by multiple paths from the signed random

walk. Generally, we perform multiple random walks on a signed graph $\mathcal{G}$, obtaining multiple random walk sequences $Q = \{q_i\}_{i=1}^{l}$ of length l, where $q_i \in \mathcal{V}$. The random walk sequence in the graph starts from an initial node q_i, the next node q_{i+1} is randomly sampled from its neighbors $\mathcal{N}(q_i)$. Additionally, to discover long-range patterns, we use a non-backtracking approach where the previous node is excluded when sampling the next node, unless it is the only neighbor available, ensuring that each node's predecessor and successor are distinct. We define a function $\sigma : e_{ij} \rightarrow \{1, -1\}$. Given a random walk sequence Q, the signed random walk distance $\psi(v_i, v_j)$ is defined as follows:

$$\psi(v_i, v_j) = \begin{cases} \left\{ \left(\prod_{u=n}^{m-1} \sigma(q_u, q_{u+1}) \right) \cdot |m - n| \middle| \right. \\ \left. q_m = v_i \wedge q_n = v_j \right\}, & \text{if} v_i, v_j \in Q \\ m + 1, & \text{otherwise} \end{cases} \tag{5}$$

where m denotes the maximum path length between two nodes. If the path length between two nodes is $m + 1$, indicating that the nodes are unreachable from each other. So after performing r random walks on graph G, each node pair can obtain r random walk distance and the random walk encoding is defined as follows:

$$b_{\psi(v_i, v_j)} = w_r \cdot \frac{1}{\psi^r(v_i, v_j)} \tag{6}$$

where w_r is a learnable parameter that represents the weight of the result obtained from the r-th random walk. Then we modify the (i, j)-element of $\tilde{A}$ further with the signed random walk encoding b as:

$$\tilde{A}_{ij} = \frac{(h_i W_Q)(h_j W_K)^T}{\sqrt{d}} + \hat{A}_{ij} + b_{\psi(v_i, v_j)} \tag{7}$$

Next, we theoretically analyzed the expressive power of our signed random walk encoding compared to shortest path encoding. The shortest path encoding for signed graphs calculates the distance between two nodes and, using balance theory, assigns a positive or negative relationship to the path.

Definition 1 (Signed graph isomorphism)**.** Two signed graphs G_1 and G_2 are isomorphic, if there exists a bijection $\phi : \mathcal{V}_{\mathcal{G}_1} \rightarrow \mathcal{V}_{\mathcal{G}_2}$, for every pair of nodes $v_1, v_2 \in V_{\mathcal{G}_1}$, $e_{ij} \in \mathcal{E}_1$, if and only if $e_{\phi(v_i), \phi(v_j)} \in \mathcal{E}_2$ and $\delta(e_{ij}) = \delta(e_{\phi(v_i), \phi(v_j)})$.

Theorem 1. *With a sufficient number of random walks, signed random walk encoding is more expressive than that based on a fixed shortest path for signed graph.*

Proof. We demonstrate our proof with two cases: 1) If two signed graphs are isomorphic under signed random walk encoding, they are also isomorphic under shortest path encoding. 2) In some cases, distinct signed graphs identified as non-isomorphic by signed random walk encoding are indistinguishable by shortest path encoding.

In case 1, given a node pair (v_i, v_j), the signed random walk distance varies depending on different random walk samples. Let $\mathcal{S}_{i,j} = \{s_1, ..., s_r\}$ denote the set of r possible distances between v_i and v_j sampled by different signed random walk. With a sufficient number of random walks, the shortest distance with a positive or negative sign can be explored in $S_{i,j}$. If two graphs G_1 and G_2 are identified as isomorphic by signed random walk encoding, then every node pair (v_i^1, v_j^1) in G_1 can find a corresponding node pair (v_i^2, v_j^2) in G_2 and corresponding $S_{i,j}^1$, $S_{i,j}^2$ are identical. Therefore, the shortest distances of the two nodes are also the same, G_1 and G_2 can also be identified as isomorphic by the shortest path encoding.

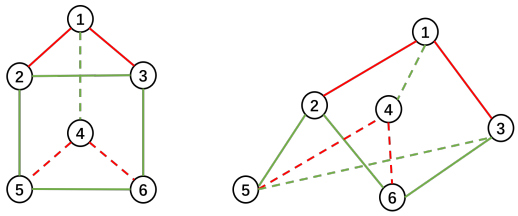


Figure 4: Encoding based on the shortest path cannot map these two graphs to different embeddings, while encoding based on the signed random walk can map them to different embeddings.

As illustrated in case 2 (Figure 4), the structures of the two graphs are different. However, the shortest path encoding cannot identify these two graphs as non-isomorphic. In the left graph, the shortest path distances from node v_1 and node v_4 to the other nodes are $\{-1, -1, 1, -2, -2\}$ and the shortest path distances from the remaining nodes to all other nodes are $\{-1, 1, 1, 2, -2\}$. For v_1 and v_4 in the right graph, the shortest path distances to the other nodes are also $\{-1, -1, 1, -2, -2\}$ and the shortest path distances to the other nodes are also $\{-1, 1, 1, 2, -2\}$ for the remaining nodes. Therefore, the shortest path encoding views them as isomorphic. In contrast, signed random walk encoding regards two graphs as non-isomorphic. For example, one can revisit the node in 3 steps in the left graph, while it is impossible on the right graph. Therefore, signed random walk encoding is more powerful than that based on a fixed shortest path. □

For the issues present in Challenge 1, we further analyzed the expressive power of our graph transformer architecture based on signed random walk encoding and SGCN-based SGNN models. Our conclusion is as follow:

Theorem 2. *(Proof in Appendix) With a sufficient number of random walk length, the graph transformer architecture based on signed random walk encoding is more expressive than SGCN.*

According to the paper (Zhang et al. 2023b), the positive or negative relationship between two nodes in an unbalanced cycle is contradictory (One path is positive, while the other is negative). Therefore, it is impossible to infer their relationship from the local structure. Signed random walk encoding combines information from multiple weighted paths to judge the relationship between nodes. This allows us to help the model obtain reasonable association information between two points within an unbalanced cycle.

Transformer Layer. In the self-attention layer, each attention head calculation formula is as follow:

$$\mathrm{Attn}(h^{(l-1)}) = \mathrm{softmax}(\tilde{A})V^{(l-1)} \tag{8}$$

We introduce layer normalization (LN) before both the multi-head self-attention (MHA) mechanism and the feed-forward neural network (FFN). Moreover, for the feed-forward network, we unify the dimensions of the input, output, and the inner layer to the same value d. The MHA concatenates the representations from each attention head and maps them into a d-dimensional vector. Then we formally characterize the Transformer layer as below:

$$\begin{aligned} h^{'(l)} &= \mathrm{MHA}(\mathrm{Attn}(h^{(l-1)})) + h^{(l-1)}, \\ h^{(l)} &= \mathrm{FFN}(\mathrm{LN}(h^{'(l)})) + h^{'(l)} \end{aligned} \tag{9}$$

The loss function follows SGCN (Derr, Ma, and Tang 2018), which can be found in the Appendix. The training algorithm of encoding module is given in Algorithm 1.

Algorithm 1: Training Algorithm of SE-SGformer

Input: A signed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}^+, \mathcal{E}^-)$; SE-SGformer model f_G; number of Transformer layers L; number of attention head a; max degree of positive or negative degrees D; number of random walks r; the length of random walk l; maximum path length m.
Output: node embedding $z_i, \forall v_i \in \mathcal{V}$.
Initialize the parameter of f_G.
Use spectral methods to generate initial node representation $x_i (\forall v_i \in \mathcal{V})$.
for $v_i \in \mathcal{V}$ **do**
  Obtain the $h_i^{(0)}$ by adding centrality encoding to node features in Equation 3.
end for
repeat
  for l = 1 to L **do**
    Obtain the adjacency matrix encoding by Equation 4 and random walk encoding by Equation 6.
    Obtain the attention score matrix $\tilde{A}$ by Equation 7.
    $h^{'(l)} = \mathrm{MHA}(\mathrm{Attn}(h^{(l-1)}))) + h^{(l-1)}$,
    $h^{(l)} = \mathrm{FFN}(\mathrm{LN}(h^{'(l)})) + h^{'(l)}$,
  end for
  Update the parameters of f_G based on loss (Equation 18).
until convergence
$z_i \leftarrow \mathbf{h_i}^{(L)}, \forall v_i \in \mathcal{V}$

Explainable Prediction Module

Intuitively, if the similarity between node v_i and node v_j is closer to the similarity between node v_j and its most similar positive neighbors, there is likely a positive relationship between node v_i and node v_j. Similarly, if the similarity between node v_i and node v_j is closer to the similarity between node v_j and its least similar negative neighbors, then there is likely a negative relationship between node v_i and node v_j. In this paper, we use Euclidean Distance to measure the similarity between nodes. The closer the distance between two nodes, the more similar they are. Let z_i represent the embedding obtained after the Transformer Encoder encoding the node v_i. Then the distance of node v_i and node v_j can be calculated as follow:

$$d_{ij} = \|z_i - z_j\|_2 \tag{10}$$

We first generate the diffusion matrix S for the entire graph through the SRWR algorithm (Jung et al. 2016). S is based on the adjacency matrix and incorporates the potential positive and negative relationship information between nodes that the SRWR has uncovered. $S_{ij} = 1$ indicates a positive edge between node v_i and node v_j, while $S_{ij} = -1$ indicates a negative edge between them and $S_{ij} = 0$ signifies no relationship between the two nodes. The specific acquisition method of the diffusion matrix S can be found in Appendix. For each node, we sample n nodes from its positive neighbors using the adjacency matrix A, calculate the distance between the node and these neighbors, sort them in ascending order, and select K-nearest positive neighbors. Similarly, we also sample n nodes from its negative neighbors, calculate the distance between the node and these neighbors, sort them in ascending order, and select K-farthest negative neighbors. If a node does not have enough K negative neighbors, we sample its negative neighbors from the diffusion matrix S.

Then we can identify the K-nearest (farthest) positive (negative) neighbor nodes of a given node. Let $\mathcal{K}_{ip} = \left\{v_{ip}^1, \ldots, v_{ip}^k\right\}$ be the set of K-nearest positive nodes of the node v_i and $\mathcal{K}_{in} = \left\{v_{in}^1, \ldots, v_{in}^k\right\}$ be the set of K-farthest negative nodes of the node v_i. d_{ip} denotes the median distance from v_i to its K-nearest positive nodes:

$$d_{ip} = \text{median}\left(\|z_i - z_j\|_2, v_j \in \mathcal{K}_{ip}\right) \tag{11}$$

Similarly, d_{in} denotes the median distance from v_i to its K-farthest negative nodes:

$$d_{in} = \text{median}\left(\|z_i - z_j\|_2, v_j \in \mathcal{K}_{in}\right) \tag{12}$$

Then d_{ij} denotes the distance from v_i to v_j which is calculated by Equation 10. The result of link sign prediction can be obtained by comparing the distances between d_{ij} and d_{ip} or d_{in}. If d_{ij} is closer to d_{ip}, the result of link sign prediction $\hat{y}_{ij} = 1$ or if d_{ij} is closer to d_{in}, then $\hat{y}_{ij} = -1$.

Time complexity analysis. The time complexity of Transformer primarily comes from the self-attention mechanism, which is $(O(|V|^2 \cdot d))$. The time complexity of the discriminate function can be analyzed by considering the key operations it performs. For each node, the function calculates the distances between the node's embedding and its neighbors' embeddings. For positive neighbors, it computes the distance for n sampled neighbors with a time complexity of $O(n \cdot d)$. A similar calculation is performed for negative neighbors. Subsequently, the function selects the top K nearest neighbors from the n samples, which has a complexity of $O(K \log n)$. These steps are repeated for each of the $|V|$ nodes, leading to a total complexity of $O(|V| \cdot (n \cdot d + K \log n))$. The overall time complexity is $O(|V| \cdot (n \cdot d + K \log n)) + |V|^2 \cdot d)$.

Experiments

In this section, we conduct experiments on real-world datasets to verify the effectiveness of SE-SGformer. In particular, we aim to answer the following research questions:

- **RQ1**: Can SE-SGformer provide accurate predictions and explanatory information ?

Dataset	GCN	GAT	SGCN	SNEA	SGCL	SIGformer	SE-SGformer
Amazon-music	63.87 ± 3.57	65.39 ± 2.88	70.63 ± 0.69	70.48 ± 0.05	78.26 ± 1.52	58.64 ± 0.64	**79.20 ± 0.23**†
Epinions	71.07 ± 1.26	73.97 ± 1.64	**86.97 ± 1.53**	82.26 ± 0.57	70.83 ± 5.10	57.07 ± 0.38	72.84 ± 1.78
KuaiRand	44.35 ± 0.00	51.63 ± 2.84	**62.85 ± 0.05**	61.95 ± 0.13	60.68 ± 0.99	61.40 ± 0.47	56.89 ± 0.12
KuaiRec	61.56 ± 0.42	65.73 ± 0.74	85.11 ± 0.11	79.69 ± 0.01	79.84 ± 3.13	61.31 ± 1.37	**85.60 ± 0.05**†
WikiRfa	70.79 ± 6.02	71.31 ± 3.56	78.69 ± 1.06	75.20 ± 0.13	75.02 ± 4.33	65.60 ± 0.94	**79.99 ± 0.08**†
WikiElec	66.21 ± 1.50	66.50 ± 1.76	79.14 ± 0.48	77.10 ± 0.68	79.63 ± 2.82	65.74 ± 2.72	**80.63 ± 0.08**†
Bitcoin-OTC	83.77 ± 0.60	86.37 ± 1.24	88.22 ± 0.69	86.05 ± 0.46	87.65 ± 1.74	80.30 ± 2.32	**90.03 ± 0.35**‡
Bitcoin-Alpha	83.99 ± 1.61	86.25 ± 0.38	87.96 ± 0.38	88.95 ± 0.15	83.01 ± 3.79	73.82 ± 3.55	**89.88 ± 0.40**‡

Table 1: Comparison of Accuracy(%) across Different Models. The best scores are in bold, and the second-best ones are underlined. "†" and "‡" indicate the statistically significant improvements with $p < 0.05$ and $p < 0.01$ (one-sided paired t-test) over the best baseline, respectively.

- **RQ2**: How do the hyper-parameters affect the performance of SE-SGformer ?
- **RQ3**: How does each component of SE-SGformer contribute to the link sign prediction performance?

Our experimental datasets include Bitcoin-OTC, Bitcoin-Alpha, WikiElec, WikiRfa, Epinions, KuaiRand, KuaiRec and Amazon-music, with baseline methods being GCN (Kipf and Welling 2016), GAT (Veličković et al. 2017), SGCN (Derr, Ma, and Tang 2018), SNEA (Li et al. 2020), SGCL (Shu et al. 2021), and SIGFormer (Chen et al. 2024). For detailed information on the datasets and baselines, please refer to the Appendix.

Metrics. Prediction accuracy and Explanation accuracy (Precision@K) are two metrics for evaluating the performance of link sign prediction and explanation performance. Prediction accuracy measures the overall correctness of the model in predicting whether an edge is positive or negative by calculating the proportion of correctly predicted edges. Also, we generate corresponding explanatory information for real-world datasets, which includes the K-nearest positive neighbors and K-farthest negative neighbors for each node as the ground truth for explanation. The specific process of generation is detailed in Appendix. Then precision@K is the proportion of the nodes identified after sorting the neighbors of the nodes during the decision-making process, which constitutes the explanatory truth.

Configurations. All experiments were conducted on a 64-bit machine equipped with two NVIDIA GPUs (NVIDIA L20, 1440 MHz, 48 GB memory). For our SE-SGformer model, we used the Adam optimizer and performed a grid search to determine the hyperparameters. Specifically, we set the hidden embedding dimension d to 128, the learning rate to 1×10^{-3}, the weight decay to 5×10^{-4}, and the number of Transformer layers to $L = 1$. For the discriminator, we choose $K = 40$ and the number of randomly sampled neighbors $m = 200$. We searched for the optimal L in the range [1, 4] with a step size of 1, d in the range [16, 32, 64, 128], and max degree in the range [6, 8, 10, 12, 14].

Performance and Explanation Quality (RQ1)

To answer RQ1, we compare the performance of SE-SGformer with baselines on real-world datasets in terms of link sign prediction. All datasets were experimented with five times, and the link sign prediction accuracy and standard deviation are shown in Table 1. From the table, we can observe the following results:

- Our method outperforms SGCN, SNEA, and SGCL on most datasets, indicating that our encoding approach produces suitable graph representations and, combined with our explainable decisions, achieves excellent results. As mentioned earlier, our method can alleviate issues such as the scarcity of negative edges in signed graphs and the inability to learn appropriate representations for unbalanced cycles. Therefore, our method can achieve better performance.
- The performance of SE-SGformer far surpasses that of GCN, GAT, that is because our method makes better use of the information from negative edges compared to unsigned GNNs.
- SIGformer is a relatively new signed graph Transformer model for recommendation systems, and our model outperforms SIGformer on most datasets. This also demonstrates the effectiveness of our unique encoding design.

Then, we evaluate the quality of the explanatory information of the k- nearest positive (farthest negative) neighbors identified in the explainable decision-making process. We set K = 40 and compared our method with the baseline on three datasets. The precise@40 and standard deviation are shown in Table 2. We can observe that our model achieves better explanatory accuracy compared to other methods, indicating that it performs well in identifying K-nearest positive (K-farthest negative) neighbors. This also reflects that our encoder is capable of learning suitable graph representations. The reason we set $K = 40$ is that a node typically has many positive neighbors, hence a small value of K could lead to the selection of nodes that are not representative, resulting in more errors. Also, as mentioned earlier, there are not many negative neighbors in the actual dataset, and our model can still achieve good explanatory accuracy with $K = 40$. This indicates that our diffusion matrix effectively mitigates the issue of fewer negative edges in the datasets.

Hyperparameter Sensitivity Analysis (RQ2)

In this subsection, we conduct a detailed sensitivity analysis of the key hyper-parameters max degree (D), dim (d), and layer (L). D represents the maximum value of positive or

Model	Bitcoin-OTC	Bitcoin-Alpha	Amazon-music
SGCN	57.25 ± 0.15	54.88 ± 0.14	60.78 ± 0.07
SNEA	55.09 ± 0.16	55.43 ± 0.15	60.29 ± 0.07
SGCL	54.10 ± 0.16	54.59 ± 0.15	61.70 ± 0.08
SIGformer	53.33 ± 0.14	51.27 ± 0.09	58.76 ± 0.05
SE-SGformer	**75.19 ± 0.13**	**94.47 ± 0.58**	**76.07 ± 0.20**

Table 2: The metric precision@40 (%) of baselines on different datasets

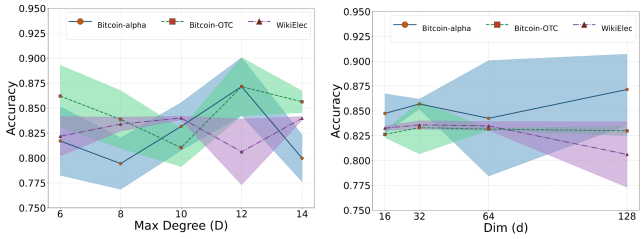


Figure 5: Parameter sensitivity analysis

negative degrees, d refers to the dimension of the node embedding, and L indicates the number of Transformer layers. We systematically vary these hyperparameters to assess their impact on the model's performance. In Figure 5, $D$is varied between 6 and 14, showing that while changes in D slightly influence the model's performance, the effect is manifested as minor fluctuations across different datasets. d is adjusted from 16 to 128 to explore how the dimension of the node embedding affects performance, with results indicating relatively stable performance and only slight deviations at larger dimensions. Lastly, L is varied from 1 to 4, revealing that increasing the number of layers generally reduces the model's performance, detailed experimental results can be found in Appendix.

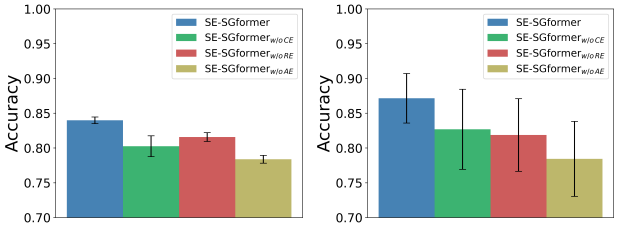


Figure 6: Ablation study on Bitcoin-OTC (left) and Bitcoin-Alpha (right) datasets.

Ablation Study (RQ3)

To answer RQ3, we conducted ablation experiments to explore the impact of the three encodings of Transformer on prediction accuracy. SE-SGformer$_{\text{w/o-CE}}$ denotes the variant without centrality encoding. SE-SGformer$_{\text{w/o-RE}}$ denotes the variant without signed random walk encoding. SE-SGformer$_{\text{w/o-AE}}$ denotes the variant without adjacency matrix encoding. The experiment results on Bitcoin-OTC and Bitcoin-Alpha datasets are reported in Figure 6. We can observe that the model's performance consistently decreases when each of the three types of encoding is removed, i.e., SE-SGformer$_{\text{w/o-CE}}$, SE-SGformer$_{\text{w/o-RE}}$ and SE-SGformer$_{\text{w/o-AE}}$ exhibit significantly inferior performance than SE-SGformer on Bitcoin-OTC and Bitcoin-Alpha. This result clearly reflects the effectiveness of the three types of encoding we used. Among them, the model's performance drops the most when the adjacency matrix encoding is removed, indicating that the information about the direct relationships between nodes and their first-order neighbors is highly useful.

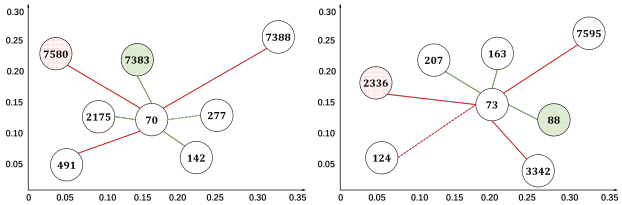


Figure 7: Case study for node pairs with positive link (left) and negative link (right). Green lines represent positive edges, red lines represent negative edges, and dashed lines indicate the edges to be predicted.

Case Study. We conduct a case study using the Bitcoin-Alpha dataset to illustrate how our discriminator accurately determines the sign of an edge by identifying k neighbors. Specifically, we analyze a pair of points where the ground truth of their edge is positive, showcasing this as a typical case for predicting a positive edge (Figure 7 (left)). In the figure, we observe the median distance of three positive neighbors (e.g., point 7383) and the median distance of three negative neighbors (e.g., point 7580) for point 70. The distance between point 70 and point 277 is evidently closer to the median distance of the positive neighbors, leading to the prediction of this relationship as a positive edge. Similarly, we identified a typical case for predicting a negative edge (Figure 7 (right)). The distance between point 73 and point 124 is closer to the median distance of the negative neighbors (e.g., point 2336), resulting in the prediction of a negative edge.

Conclusion

In this paper, we address the challenge of self-explainable SGNNs by proposing SE-SGformer, a novel graph Transformer-based model for signed graphs. Our model predicts link signs by identifying K-nearest positive and K-farthest negative neighbors. We also theoretically prove that our signed random walk encoding is more powerful than the shortest path encoding and our Transformer architecture with signed random walk encoding is more powerful than SGCN. Extensive experiments on real-world datasets validate our model's effectiveness. As the first exploration into the explainability of link sign prediction, we anticipate further research in this area.

References

Baldassarre, F.; and Azizpour, H. 2019. Explainability techniques for graph convolutional networks. *arXiv preprint arXiv:1905.13686*.

Bonchi, F.; Galimberti, E.; Gionis, A.; Ordozgoiti, B.; and Ruffo, G. 2019. Discovering polarized communities in signed networks. In *Proceedings of the 28th acm international conference on information and knowledge management*, 961–970.

Chen, D.; O'Bray, L.; and Borgwardt, K. 2022. Structure-aware transformer for graph representation learning. In *International Conference on Machine Learning*, 3469–3489. PMLR.

Chen, S.; Chen, J.; Zhou, S.; Wang, B.; Han, S.; Su, C.; Yuan, Y.; and Wang, C. 2024. SIGformer: Sign-aware Graph Transformer for Recommendation. *arXiv preprint arXiv:2404.11982*.

Chen, Y.; Qian, T.; Liu, H.; and Sun, K. 2018. " Bridge" Enhanced Signed Directed Network Embedding. In *Proceedings of the 27th acm international conference on information and knowledge management*, 773–782.

Dai, E.; and Wang, S. 2021. Towards self-explainable graph neural network. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 302–311.

Deng, J.; and Shen, Y. 2024. Self-Interpretable Graph Learning with Sufficient and Necessary Explanations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 11749–11756.

Derr, T.; Ma, Y.; and Tang, J. 2018. Signed graph convolutional networks. In *2018 IEEE International Conference on Data Mining (ICDM)*, 929–934. IEEE.

Huang, J.; Shen, H.; Hou, L.; and Cheng, X. 2019. Signed graph attention networks. In *International Conference on Artificial Neural Networks*, 566–577. Springer.

Huang, J.; Shen, H.; Hou, L.; and Cheng, X. 2021. SDGNN: Learning Node Representation for Signed Directed Networks. *arXiv preprint arXiv:2101.02390*.

Javari, A.; Derr, T.; Esmailian, P.; Tang, J.; and Chang, K. C.-C. 2020. Rose: Role-based signed network embedding. In *Proceedings of The Web Conference 2020*, 2782–2788.

Jung, J.; Jin, W.; Sael, L.; and Kang, U. 2016. Personalized ranking in signed networks using signed random walk with restart. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 973–978. IEEE.

Jung, J.; Yoo, J.; and Kang, U. 2020. Signed Graph Diffusion Network. *arXiv preprint arXiv:2012.14191*.

Kim, J.; Park, H.; Lee, J.-E.; and Kang, U. 2018. Side: representation learning in signed directed networks. In *Proceedings of the 2018 World Wide Web Conference*, 509–518.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Li, Y.; Tian, Y.; Zhang, J.; and Chang, Y. 2020. Learning signed network embedding via graph attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 4772–4779.

Luo, D.; Cheng, W.; Xu, D.; Yu, W.; Zong, B.; Chen, H.; and Zhang, X. 2020. Parameterized explainer for graph neural network. *Advances in neural information processing systems*, 33: 19620–19631.

Ni, L.; Wang, S.; Zhang, Z.; Li, X.; Zheng, X.; Denny, P.; and Liu, J. 2024. Enhancing student performance prediction on learnersourced questions with sgnn-llm synergy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 23232–23240.

Pope, P. E.; Kolouri, S.; Rostami, M.; Martin, C. E.; and Hoffmann, H. 2019. Explainability methods for graph convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10772–10781.

Rong, Y.; Bian, Y.; Xu, T.; Xie, W.; Wei, Y.; Huang, W.; and Huang, J. 2020. Self-supervised graph transformer on large-scale molecular data. *Advances in neural information processing systems*, 33: 12559–12571.

Schnake, T.; Eberle, O.; Lederer, J.; Nakajima, S.; Schütt, K. T.; Müller, K.-R.; and Montavon, G. 2021. Higher-order explanations of graph neural networks via relevant walks. *IEEE transactions on pattern analysis and machine intelligence*, 44(11): 7581–7596.

Seo, S.; Kim, S.; and Park, C. 2024. Interpretable prototype-based graph information bottleneck. *Advances in Neural Information Processing Systems*, 36.

Shu, L.; Du, E.; Chang, Y.; Chen, C.; Zheng, Z.; Xing, X.; and Shen, S. 2021. SGCL: Contrastive Representation Learning for Signed Graphs. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 1671–1680.

Tang, J.; Aggarwal, C.; and Liu, H. 2016. Node classification in signed social networks. In *Proceedings of the 2016 SIAM international conference on data mining*, 54–62. SIAM.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Vu, M.; and Thai, M. T. 2020. Pgm-explainer: Probabilistic graphical model explanations for graph neural networks. *Advances in neural information processing systems*, 33: 12225–12235.

Wang, H.; Zhang, F.; Hou, M.; Xie, X.; Guo, M.; and Liu, Q. 2018. Shine: Signed heterogeneous information network embedding for sentiment link prediction. In *Proceedings of the eleventh ACM international conference on web search and data mining*, 592–600.

Ying, C.; Cai, T.; Luo, S.; Zheng, S.; Ke, G.; He, D.; Shen, Y.; and Liu, T.-Y. 2021. Do transformers really perform badly for graph representation? *Advances in neural information processing systems*, 34: 28877–28888.

Ying, Z.; Bourgeois, D.; You, J.; Zitnik, M.; and Leskovec, J. 2019. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32.

Yu, J.; Xu, T.; Rong, Y.; Bian, Y.; Huang, J.; and He, R. 2020. Graph information bottleneck for subgraph recognition. *arXiv preprint arXiv:2010.05563*.

Yuan, H.; Tang, J.; Hu, X.; and Ji, S. 2020. Xgnn: Towards model-level explanations of graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 430–438.

Yuan, H.; Yu, H.; Gui, S.; and Ji, S. 2022. Explainability in graph neural networks: A taxonomic survey. *IEEE transactions on pattern analysis and machine intelligence*, 45(5): 5782–5799.

Yuan, S.; Wu, X.; and Xiang, Y. 2017. SNE: signed network embedding. In *Pacific-Asia conference on knowledge discovery and data mining*, 183–195. Springer.

Zhang, H.; Wu, B.; Yuan, X.; Pan, S.; Tong, H.; and Pei, J. 2024. Trustworthy graph neural networks: aspects, methods, and trends. *Proceedings of the IEEE*.

Zhang, Z.; Liu, J.; Zhao, K.; Yang, S.; Zheng, X.; and Wang, Y. 2023a. Contrastive Learning for Signed Bipartite Graphs. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1629–1638.

Zhang, Z.; Liu, J.; Zheng, X.; Wang, Y.; Han, P.; Wang, Y.; Zhao, K.; and Zhang, Z. 2023b. RSGNN: A Model-agnostic Approach for Enhancing the Robustness of Signed Graph Neural Networks. In *Proceedings of the ACM Web Conference 2023*, 60–70.

Zhang, Z.; Liu, Q.; Wang, H.; Lu, C.; and Lee, C. 2022. Protgnn: Towards self-explaining graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 9127–9135.

Zhu, W.; Wen, T.; Song, G.; Wang, L.; and Zheng, B. 2023. On structural expressive power of graph transformers. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 3628–3637.

Reproducibility Checklist

This paper:

- Includes a conceptual outline and/or pseudocode description of AI methods introduced (**yes**)
- Clearly delineates statements that are opinions, hypothesis, and speculation from objective facts and results (**yes**)
- Provides well marked pedagogical references for less-familiare readers to gain background necessary to replicate the paper (**yes**)

Does this paper make theoretical contributions? (**yes**)
If yes, please complete the list below.

- All assumptions and restrictions are stated clearly and formally. (**yes**)
- All novel claims are stated formally (e.g., in theorem statements). (**yes**)
- Proofs of all novel claims are included. (**yes**)
- Proof sketches or intuitions are given for complex and/or novel results. (**yes**)
- Appropriate citations to theoretical tools used are given. (**yes**)
- All theoretical claims are demonstrated empirically to hold. (**yes**)
- All experimental code used to eliminate or disprove claims is included. (**yes**)

Does this paper rely on one or more datasets? (**yes**)
If yes, please complete the list below

- A motivation is given for why the experiments are conducted on the selected datasets (**yes**)
- All novel datasets introduced in this paper are included in a data appendix. (**yes**)
- All novel datasets introduced in this paper will be made publicly available upon publication of the paper with a license that allows free usage for research purposes. (**NA**)
- All datasets drawn from the existing literature (potentially including authors' own previously published work) are accompanied by appropriate citations. (**yes**)
- All datasets drawn from the existing literature (potentially including authors' own previously published work) are publicly available. (**yes**)
- All datasets that are not publicly available are described in detail, with explanation why publicly available alternatives are not scientifically satisficing. (**NA**)

Does this paper include computational experiments? (**yes**)
If yes, please complete the list below.

- Any code required for pre-processing data is included in the appendix. (**yes**).
- All source code required for conducting and analyzing the experiments is included in a code appendix. (**yes**)
- All source code required for conducting and analyzing the experiments will be made publicly available upon publication of the paper with a license that allows free usage for research purposes. (**yes**)
- All source code implementing new methods have comments detailing the implementation, with references to the paper where each step comes from (**partial**)
- If an algorithm depends on randomness, then the method used for setting seeds is described in a way sufficient to allow replication of results. (**yes**)
- This paper specifies the computing infrastructure used for running experiments (hardware and software), including GPU/CPU models; amount of memory; operating system; names and versions of relevant software libraries and frameworks. (**yes**)
- This paper formally describes evaluation metrics used and explains the motivation for choosing these metrics. (**yes**)

- This paper states the number of algorithm runs used to compute each reported result. (**yes**)
- Analysis of experiments goes beyond single-dimensional summaries of performance (e.g., average; median) to include measures of variation, confidence, or other distributional information. (**yes**)
- The significance of any improvement or decrease in performance is judged using appropriate statistical tests (e.g., Wilcoxon signed-rank). (yes/partial/no)
- This paper lists all final (hyper-)parameters used for each model/algorithm in the paper's experiments. (**yes**)
- This paper states the number and range of values tried2 per (hyper-) parameter during development of the paper, along with the criterion used for selecting the final parameter setting. (**yes**)