Anomaly Detection in Dynamic Graphs using Multiple Encoding Strategies via Transformers

Vishal Krishna Singh, Niharika Anand, Amrit Pal, Abishi Chowdhury and Arjun Srivastava

***Abstract*—Accurate anomaly detection in dynamic graph networks suffers due to lack of coverage of all aspects of information; specifically temporal, spatial and centrality based cross-coupled information. This work aims to address the challenge of precise and accurate anomaly detection in dynamic graph networks. It uses a graph-based diffusion technique to sample a fixed-size, yet cross-coupled, information-rich circumstantial node set for target edges. Centrality enabled spatial-temporal node encoding is considered as input to the dynamic graph based transformer network. The proposed method uses a set of four elements to make up the node encoding. The four encoding terms are combined to create an input that contains extensive centrality based cross-coupled spatial-temporal node encoding. The transformer module simultaneously captures all the required attributes with a single encoder. The performance of the proposed method is validated on six different datasets; UCI Messages, Bitcoin-Alpha, Digg Social, Enron Email, Epinions-Trust and AS-Topology. The proposed method outperforms the existing methods in terms of *AUC-ROC score*, *accuracy*, *loss*, and *precision*. Results show an improvement of 2.42% AUC-ROC value over the existing methods proving the models ability to counter over-fitting and provide accurate results.**

***Index Terms*—Anomaly Detection, Dynamic Graphs, Edge Networks, Transformers, Input Embeddings.**

I. Introduction

Dynamic graphs are data structures where attributes change over time and are therefore also referred to as temporal graphs [1] [2]. Recent years have seen dynamic graphs being used in several applications including modeling systems like social media networks, traffic flow networks and designing routing networks [3], [4]. Despite a wide range of applications, the dynamic nature of these networks imposes several inherent constraints on performance in terms of data management, anomaly detection and feature optimization among others [4]. For instance, anomalies in dynamic graphs can lead to surprising shifts in community structures or clustering, as well as sharp increases or decreases in connectivity. They may also cause abrupt changes in node properties or behavior, unexpected variations in centrality measures, and deviations from previous interaction patterns [5]. This work is aimed at accurately detecting anomalies in dynamic graphs, which may arise due to various factors such as rapid shifts in network activity, changing node associations or the emergence of novel patterns that deviate from expected temporal trends [5], [6], [7]. Detecting such anomalies as early as possible is essential to ensure timely and effective responses.

The temporal correlation of quickly changing variables must be considered when designing solutions for accurate anomaly detection. Furthermore, while processing large volumes of dynamic graph data, the performance is hindered by high false positives, varying network size, and resource-hungry computations. To demonstrate how linked information influences the detection of edge abnormalities, a typical dynamic graph network is shown in Fig. 1. The three graphs represent a section of dynamic graph stream at sequential times T^{t-2}, T^{t-1} and T^{t}. The black lines represent normal edges between nodes, while the red line at t represents an anomalous edge. The dark grey colored node represents newly added node in that specific timestamp. Different colors are used to denote the corresponding nodes in the previous timestamps as per the sliding window selection, taking the illustrative window size as 3 nodes. Given that the network connects the neighbors in the previous timestamps, the black edge is typical. The two green nodes consistently have a distance from one another in the earlier snapshots, which means that the edge between these two green nodes, i.e. red might be an anomalous edge. Thus, it is evident that shared neighborhoods and previous interactions (examples of structural aspects) need to be taken into account simultaneously while making judgments.

In the past, a variety of methods have been proposed to address the issue of accurate anomaly detection in dynamic graphs. The shallow learning processes, based on structural connection model or historical behavior analysis, such as GOutlier [8] and CM-Sketch [9], are some of the popular techniques. In recent times, several deep learning (DL) based techniques have shown effectiveness for time-variant graphs as an unique approach. For instance, NetWalk [10] uses cluster-based dynamic deep graph embedding to identify deviations; AddGraph [9], StrGNN [11], and Cu-BLSTM [12] also make use of such deep neural network models to address the issue. Another interesting approach to address this issue is the use of transformers for extracting global dependencies in data. In contrast to the conventional architectures that processes data sequentially, transformers are neural network architectures centered on extracting global dependencies in data through processing the full input concurrently [13]. To effectively learn complicated patterns, transformers depend on self-attention systems that weigh various input components differently during processing. Due to their effectiveness in handling long-distance dependencies, transformers have become increasingly

Corresponding author: Vishal Krishna Singh (email: v.k.singh@essex.ac.uk)

Vishal Krishna Singh is with School of Computer Science and Electronics Engineering, University of Essex, Colchester Campus. Wivenhoe Park, Colchester, CO4 3SQ, United Kingdom.

Niharika Anand is with Department of Information Technology, Indian Institute of Information Technology, Lucknow, 226002, Uttar Pradesh, India.

Amrit Pal and Abishi Chowdhury are with School of Computer Science and Engineering, Vellore Institute of Technology, Chennai, 600127, Tamil Nadu, India.

Arjun Srivastava is with the Department of Computer Science, Indian Institute of Information Technology, Lucknow, 226002, Uttar Pradesh, India.

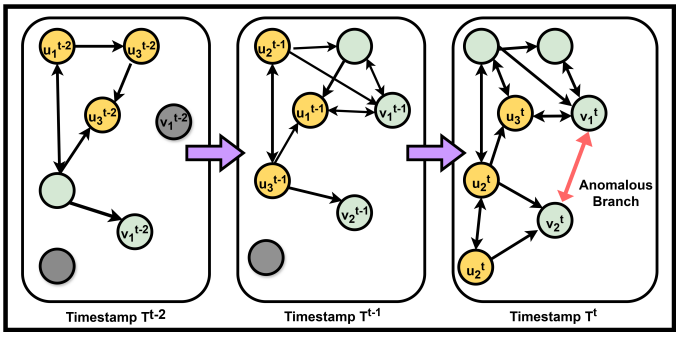


Fig. 1: Impact of coupled information on appropriateness of abnormal edge detection with illustrative window size as 3.

useful in applications such as graphs and sequence modeling.

Transformers improve the detection of dynamic network anomalies by modeling complicated temporal dependencies and capturing long-distance linkages between changing graph variables. The information is processed holistically using a self-attention mechanism, which is essential for evaluating the behavior of dynamic graphs. An added advantage of transformers is their ability to spot abnormalities that span several time points and display complex temporal patterns. However, handling the sparsity and inconsistency of dynamic graph data requires in-network optimizations. These optimizations are aimed at overcoming the issues of lack of raw information, discriminative knowledge learning and over-fitting. To clarify, the lack of raw information for dynamic graph embedding limits the ability to capture temporal patterns and extract valuable features. This, in turn, makes it difficult to effectively represent evolving network dynamics for accurate anomaly identification. In transformer-based anomaly detection for dynamic graphs, one key challenge is learning discriminatory knowledge effectively. This involves combining discriminative information in a way that distinguishes typical patterns from anomalous ones, while accounting for the dynamic nature of the graph data. Finally, robust generalization and accurate anomaly identification in evolving graph structures are also required. This presents the challenge of balancing model complexity with the available data, thereby increasing the risk of overfitting.

Thus, to achieve accurate anomaly detection in dynamic graphs, the proposed work addresses key challenges such as the lack of raw information, overfitting, and the need for dynamic and discriminative model learning. It employs a multiple embedding strategy combined with a dynamic graph transformer to develop an efficient anomaly detector based on binary cross-entropy. Centrality enabled spatial-temporal node encoding is considered as input to the dynamic graph based transformer network. A set of four elements make up the proposed node encoding strategy. The four encoding terms are combined to create an input that contains extensive centrality based cross-coupled spatial-temporal node encoding. In this way, the transformer simultaneously captures all the required attributes with a single encoder. The performance of the proposed method is validated on six different datasets, UCI Messages [14], Bitcoin-Alpha [15], Digg Social [16], Enron Email [17], Epinions-Trust [18] and AS-Topology [19]. The novel contributions of the work are identified as below:

1) A graph-based diffusion technique for sampling a fixed-size yet cross-coupled information containing circumstantial node set for target edges.
2) A novel cross-coupled multiple embedding schemes considering time, space and centrality altogether is proposed to optimize the detection accuracy.
3) A detailed ablation analysis to examine how each element of the spatial, temporal and centrality node encoding contributes to the overall performance of the proposed model.
4) A mathematical analysis of the features and advantages of the proposed model.

The rest of the paper is organized as follows: Section II of the paper covers the related research. In Section III, the problem formulation is presented followed by the overall workflow and each element of the proposed framework in Section IV. Section VII presents the dataset description and experimental setup. A comprehensive sensitivity analysis of the proposed method is presented in Section VIII which precedes the Section IX where the performance analysis and results are described. Finally in Section X, the concluding comments and future directions are presented.

II. Related Studies

A. Graph based Anomaly Detection

In recent years, various anomaly detection techniques have been proposed for dynamic graphs. To identify outliers within graph streams, for instance, GOutlier [8] uses a structural connectivity paradigm. It then creates a dynamic network division to preserve connectivity behavior model. To distinguish the anomalous quality of the edge, CM-Sketch [9] takes into account both previous behavior and the surrounding structural information. StreamSpot [20] is a cluster-based technique that makes use of the traits of graph streams. It uses a centroid-based clustering technique and a unique similarity function for assorted graph attribute comparison. To ensure a significant mapping distance between irregular and normal graph snapshots in the sketch space, SpotLight [21] adopts a randomized drawing ability. These techniques fall under the category of shallow learning-based techniques since they make use of shallow mechanisms to identify anomalous edges.

A further strategy of approach uses DL techniques to identify abnormal data in dynamic graphs. In order to create node embeddings to clique embedding objectives, NetWalk [10] uses a random walk-based encoder. Dynamically updating reservoirs, are then used to describe the network's evolution. The anomaly at each edge is then scored using an evolving clustering-based anomaly analyzer. AddGraph [22] builds an end-to-end neural network framework to capture the temporal and spatial characteristics of dynamic graphs. A GRU-attention mechanism [23] is intended to integrate short- and long-term dynamic changing features, while a Graph Convolution Network (GCN) functions as structural features extractor. Using stacked GCN and the GRU, StrGNN [11] recovers the h-hop enclosed sub-graph of edges and captures the

temporal and geographical information. The learning model learns via negative sampling using a "context-dependent" noise distribution in a complete manner. Using DL with a Software-Defined Networking (SDN) architecture, the authors in [12], propose a IDS to identify different attack types in a smart consumer electronic network. In other methods, the seminal work by the authors in [24], targets the lack of informative encoding for unattributed nodes in the existing methods. The authors propose a transformer-based anomaly detection framework exploiting the spatial-temporal relations in dynamic graphs. Node encoding and transformers are used to captures informative representation from dynamic graphs for accurate anomaly detection. The authors in [25] and [26] employ random-walk and skip-gram based approach for node embedding. Both the methods use an optimized K-means clustering scheme for efficient anomaly detection in dynamically changing node relations. The authors in [27] focus on the significantly untouched issue of lack of real tags for accurate anomaly detection. The issue is addressed through an unsupervised learning framework, aimed at optimizing the graph contrastive learning module. A network reconstruction method is proposed to track and identify the anomalies in the network and uses the topological structure along with the node attributes for performance optimization. Another important method is proposed in [28]. The authors use a graph-based clustering method to divide the nodes into groups by using the eigenvectors of Laplacian matrix. By using the graph's spectrum decomposition, the proposed method obtains inherent structures and highlights the outliers.

Despite having several advantages, these approaches have notable limitations. For example, the GOutlier [8] and CM-Sketch [9], struggle with capturing complex temporal and structural dependencies in dynamic graphs which restricts their usage only to small scale datasets. On the other hand, deep learning-based methodologies, such as AddGraph [22] and StrGNN [11], have high computational costs and reliance on extensive labeled data for training.

B. Graph Embeddings and Transformers

Transformers are specially curated neural networks for learning representative embeddings for a range of input only by using attention mechanisms. First introduced in [13], the transformer model initially focused on natural language processing (NLP) machine translation problems. By adding the pre-training method, BERT [29] extends the use of transformers to a variety of deep learning applications. Numerous alternative works have been proposed since, achieving cutting-edge results on diverse NLP problems. The transformer framework has recently been adapted for computer vision (CV) [13]. Enabling pixel-wise picture segmentation, SETR [30], for example, uses multi-level feature accumulation module and a ViT-like encoder with feature extraction.Transformers are also introduced to the area of graph machine learning in several recent research. GTN [31] uses meta-path-based relationship learning on diverse graphs with transformers. HGT [32] represents a transformer model that achieves pioneering performance on variety of downstream operations for representation learning over web-scale assorted graphs. In order to get a representation over graph data, GROVER [33] incorporates the message carrying technique into the transformer design. Graph-BERT [34] builds a network model with static graph learning which is similar to BERT and presents a number of well crafted challenges for self-supervised framework pre-training. One of the recent work in [35] uses spatial-temporal embedding based auto-encoder infrastructure for outlier detection. The proposed method focuses on noisy label learning and optimizes the sensitivity parameter of the model for improved results.

However, owing to the inherent reliance on attention mechanisms, transformers tend to have high computational cost making such approaches resource-intensive, particularly for large-scale datasets in both NLP and CV [13]. The requirement of extensive pre-training and fine-tuning in models like BERT [29] and Graph-BERT [34] makes them unsuitable for several resource constrained real-world applications. Additionally, while frameworks like HGT [32] and GROVER [33] have shown success in graph machine learning tasks, their performance relies heavily on well-crafted input features. This making them less generalizable to heterogeneous graph data. The literature supports the fact that the adaptation of transformers to graphs, in methods such as GTN [31], struggles to give the desired results due to the issues of scalability and overfitting, particularly when dealing with sparse or highly dynamic graphs.

III. Problem Formulation

A. Problem Definition 1:

Consider a time variant graph, provided the number of timestamps T, it can be considered as a graph stream and is depicted as $W = \{D^t\}_{t=1}^{T}$, in which $D^t = \{V_t, E_t\}$ is considered as a graph time-frame at the provided timestamp T. The proposed work creates a link as $e_{i,j}^t = (p_i^t, p_j^t) \in E^t$ depicting an edge among nodes p_i^t and p_j^t at the time T, where $p_i^t, p_j^t \in V^t$. The proposed model denotes $a^t = \mid V^t \mid$ and $b^t = \mid E^t \mid$ representing the count of vertices and count of edges at time T. D^t is specifically mentioned in form of a binary adjacency matrix which in the proposed methodology is represented as $M^t, M^t \in R^{n^t \times n^t}$ where $M_{i,j}^t = 1$ if a connection among the nodes p_i and p_j exists at a particular time-frame T, or else $M_{i,j}^t = 0$. The problem is to identify the edges which are anomalous in each of the timestamp, making it a classification-oriented scoring problem, i.e., it needs to be checked for $M_{i,j}^t$ values which are 1, whether or not they belong to the edges E^t in that specific timestamp T.

B. Problem Definition 2:

For the time dependent graph $W = \{D^t\}_{t=1}^{T}$, the major objective of the proposed anomaly detection method is to develop a learnable function to generate anomaly score which predicts the degree of abnormality for a connection, to which a big score $f(e_{i,j}^t)$ represents more anomalous probability of $e_{i,j}^t$.

C. Problem Definition 3:

For the considered dynamic graph $W = \{D^t\}_{t=1}^{T}$, the defined learnable anomaly score function $f(e_{i,j}^t)$ is supposed to minimize over-fitting of edges $e_{i,j}^t$ such that the anomaly score prediction can be maximized, i.e. $f(e_{i,j}^t)$, making the problem a classification-oriented scoring optimization problem.

IV. Methodology

The proposed framework consists of four major steps, namely; *Subgraph sampling based on edges*, *Node encoding based on multiple embedding strategy*, *Dynamic graph based transformer module*, and *Binary cross entropy based anomaly detector,* as shown in Fig. 2. Model training is performed in such a way that the transformer architecture is fed the anomaly scores straight away for learning. Subgraphs in the first step are sampled based on their edges to get the nodes associated in different timestamps for cross-coupled information capturing. This step majorly focus on extracting the spatial, temporal and centrality information from dynamic subgraphs based on their timestamps. The combination occurs in such a manner that a single fixed-length cumulative node encoding is produced. The transformer model then uses the node encoding produced by the centrality based spatial-temporal embeddings as input. Next, the transformer module uses a single transformer model and directs it to a pooling module to extract the overall targeted knowledge of edges. Finally, a negative sampling is performed in the binary cross-entropy based anomaly detector to produce false negative edges. This is achieved using a scoring block which utilizes binary cross-entropy loss to output anomaly scores. The proposed work extensively presents the four key parts of the methodology from Section IV-A through Section IV-D.

A. Subgraph Sampling based on Edges

Edge-based subgraph sampling involves the extraction of subsets of edges from dynamic graphs. It assists in the discovery of evolving anomalies by revealing anomalous structural patterns. It is used to magnify the receptive field to an appropriate local scale since anomalies frequently appear in the local substructures of graphs, as observed in prior research [11]. Since subgraphs are the data elements for proposed work, first step is to sample them to obtain information effectively. The Fig. 1 depicts exactly how cross-coupled information is present, and how edge based sampling is performed. Each edge acts as a center and named as target edge. The origin and objective nodes are the target nodes. Nodes surrounding the target nodes are named as contextual nodes.

With respect to contextual nodes, to effectively sample them from a particular target edge to accumulate cross-coupled contextual information is a challenge. The proposed work considers the subgraph as a static graph and proceeds. One method to achieve this is with the help of identifying the *h-hop* neighbours of target nodes, but it has drawbacks. With *h-hop* neighbours sampling, the imbalanced node degree arrangement in the authentic actual datasets would degrade the performance and efficiency of the method. Consider the case for the UCI message dataset [14]. It has a maximum degree of 255 and an average degree of 14.47. The number of *h-hop* neighbors would explode for those well-liked nodes with high degrees, leading to biased information and hence reducing efficiency. Also, sampling *h-hop* neighbours, lacks in identifying the significance of each node inside the substructure.

Note that, two target nodes can be stated as shared neighbours that help with target edge detection with a better efficiency when they have both shared and unique neighbourhood nodes. While sampling contextual nodes, this straightforward technique just treats the shared and unique neighbors equally. The proposed work uses a graph-based diffusion technique for sampling a fixed-size yet cross-coupled information containing circumstantial node set for target edges in order to deal with the restrictions. The significance of each node can be assessed for a specific target node after acquiring a global perspective of the graph structure through graph diffusion.

Initially, provided an adjacency matrix of a static graph $D \in R^{n\times m}$, the graph diffusion is defined as $L \in R^{n\times m}$ by

$$L = \sum_{x=0}^{\infty} \theta_x M^x \tag{1}$$

where $M \in R^{n\times m}$ is a vague transition matrix and θ_x is weighing measure which is simply the ratio of global to that of local context or information.

Now, in order to guarantee the convergence, some conditions must be considered which require $\sum_{x=0}^{\infty} \theta_x = 1, \theta_x \in [0,1]$ and the eigenvalues of M are bounded in $[0,1]$.

For the diffusion matrix L, row l_i represents the connection among i^{th} node and every node from a global point of view. Considering the target edge $e_{tgt} = (v_1, v_2)$, by appending the associating vectors of the target nodes

$$l_{e_{tgt}} = l_{v_1} + l_{v_2} \tag{2}$$

then top x nodes are sorted and selected with the larger values to develop the provisional node set $U_{e_{tgt}}$. After that, the bigger node set for a subgraph can be formalized as

$$L_{e_{tgt}} = \{v_1, v_2, U_{e_{tgt}}\} \tag{3}$$

Equation 3 is for single static graph. But for dynamic graph, τ times sliding window technique is implemented, where τ belongs to all the timestamps of a dynamic graph.

B. Node Encoding Based on Multiple Embedding Strategy

The unattributed nature of graphs that the proposed work uses, poses a challenge to naturally identify the relevant data to use as the input. Different to the images and attributed dataset, each data item has its own raw information. The challenge also signifies the issue of creating an informative encoding as network input. One can use the identity node encoding in place of raw node feature, which represents one-hot vector for each node, just as it is used in natural language processing (NLP). But some drawbacks are observed for identity node encoding. Absence of structural and temporal information is one of the drawbacks for the one-hot encoding technique. Also, it cannot describe the structural roles or temporal status of the nodes as it just conveys the identity of the nodes. Hence dynamic graphs

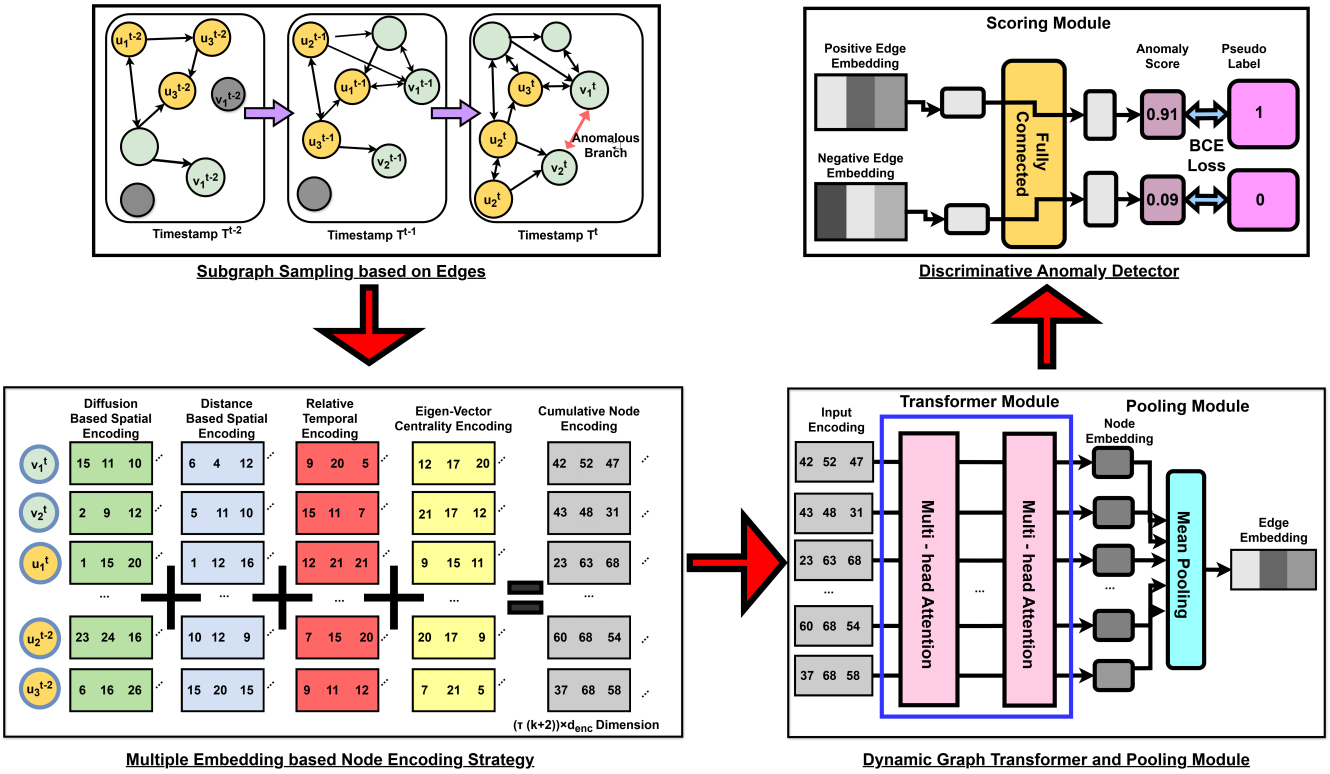


Fig. 2: The Proposed Model: *Subgraph Sampling Based on Edges*, *Node Encoding Based on Multiple Embedding Strategy*, *Dynamic Graph Based Transformer Module*, and *Binary Cross Entropy Based Anomaly Detector.*

with large amount of changing nodes is not a suitable data for identity node encoding. Additionally, changing node set is also not adapt for fixed dimension work. The proposed work presents centrality enabled spatial-temporal node encoding to be considered as input to the dynamic graph based transformer network. Four elements make up the proposed node encoding: relative temporal encoding, diffusion-dependent spatial encoding, eigen vector based centrality and distance-dependent spatial encoding.

The two terms for spatial embedding, global and local, relates to their node encoding respectively, represents the structural function of each node. Differently, the temporal encoding term provides the time specific knowledge of every constituent in the subgraph node set. Finally, the four encoding terms are combined to create an input that contains extensive centrality based cross-coupled spatial-temporal node encoding. Corresponding to the frequency-aware sin/cos functions employed in [13], the encoding is developed using learnable linear projections. The reason for this is because learnable functions may more easily describe the relationships between various timestamps or places.

1) Rank Based Diffusion Spatial Encoding: The rank based diffusion for spatial encoding is obtained as below:

$$x_{diff}(v_j^i) = linear(rank(s^i_{e_{tgt}}[index(v_j^i)])) \tag{4}$$

Where, $index(.)$ represents the enquiry function for index, $rank(.)$ represents ArgSort rank function, $linear(.)$ represents learnable linear mapping, belonging to the range $R^{d_{enc}}$ or dimension of node encoding. For every vertex in the subgraph node set $v_j^i \in S^i_{e^t_{tgt}}$, node is sorted based on diffusion value and ranking is served as a data source in the proposed work. Encoding is computed based on the ranks with a learnable linear mapping containing single layer encoding function.

2) Distance Based Spatial Encoding: The distance based spatial encoding is obtained as below:

$$x_{dist}(v_j^i) = linear(min(dist(v_j^i, v_1^i), dist(v_j^i, v_2^i))) \tag{5}$$

Where, $dist(.)$ represents relative distance computing function, $min(.)$ shows learnable linear mapping function and $min(.)$ shows minimum value function. For every node in substructure set $v_j^i \in S^i_{e^t_{tgt}}$, distance from target edge serves as information provider for encoding. This distance can be treated as the least relative distance value between two target nodes.

3) Relative Temporal Encoding: The relative temporal encoding is obtained as below:

$$x_{temp}(v_j^i) = linear(\|t - i\|) \in R^{d_{enc}} \tag{6}$$

Where $\|.\|$ is relative time computing function, which is L^2 norm in here and $linear(.)$ is learnable linear mapping. Data source here is the difference between target t and current edge i timestamps.

4) Eigenvector based Centrality Encoding: The eigenvector based centrality encoding is obtained as below:

$$x_{cent}(v_j^i) = \frac{1}{\lambda}\sum_{j=1}^{n} A_{i,j}\alpha_j \in R^{d_{enc}} \tag{7}$$

Where λ is eigenvalue associated with that particular eigenvector, and α_j represents the centrality score of that particular vertex v_i. The same equation (7) can also be represented as:

$$A \cdot x_{cent} = \Lambda \cdot x_{cent}$$

The eigenvalue centrality vector is for the largest eigenvalue. In context to this statement, the Lemma 1 is presented:

Lemma 1. *Consider a provided matrix $M \in R^{n\times n}$ containing n eigenvectors, i.e. $\{v_1, v_2, ...v_n\}$ relating to the eigenvalues $\{\Lambda_1, \Lambda_2, ...\Lambda_n\}$. Suppose v_1 is the eigenvalue with the maximum value and Λ_1 is it's corresponding eigenvector, then any random vector $r_0 \in R^{n\times 1}$ can be represented as:*

$$M^k_{r_0} \approx c \cdot v_1 \tag{8}$$

where for some constant c, $k \to \infty$

Proof. - For the linear combination $\{v_1, v_2, ...v_n\}$, i.e. independent eigenvectors, the random vector r_0 can be represented as:

$$r_0 = c_1v_1 + c_2v_2 + ... + c_mv_m$$

$$Mr_0 = c_1Mv_1 + c_2Mv_2 + ... + c_mMv_m$$

$$Mr_0 = c1\Lambda_1v_1 + c2\Lambda_2v_2 + ... + c_m\Lambda_mv_m$$

Multiplying M frequently k times on both the sides,

$$M^kr_0 = c_1\Lambda_1^kv_1 + c_2\Lambda_2^kv_2 + ... + c_m\Lambda_m^kv_m$$

$$M^kr_0 = \Lambda_1^k\left(c_1v_1 + c_2\left(\frac{\Lambda_2}{\Lambda_1}\right)^k v_2 + ... + c_m\left(\frac{\Lambda_m}{\Lambda_1}\right)^k v_m\right)$$

As $k \to \infty$ and $\left\|\frac{\Lambda_i}{\Lambda_1}\right\|^k << 1$ for $i = 1, 2, 3...m$

$$Mr_0 \approx c_1\Lambda_1^kv_1$$

Hence,

$$M^kr_0 \approx cv_1$$

Where c is a constant. □

Hence, one can assimilate that the centrality vector associated with the proposed node encoding is the one with largest eigenvalue. By Perron-Frobenius theorem, it can be stated that for M being strictly non-negative matrix, Λ exists, that is, the largest positive unique eigenvalue. Hence, eigenvector centrality of M also exists, which came out from a non-negative eigenvector.

5) Fusion of Encoding: The fusion of all the encoding strategies is obtained as below:

$$x_{total}(v_j^i) = x_{diff}(v_j^i) + x_{dist}(v_j^i) + x_{temp}(v_j^i) + x_{cent}(v_j^i) \in R^{d_{enc}} \tag{9}$$

provided the target edge e^t_{tgt}, for each node in subgraph node set, the encoding is calculated and stacked into encoding matrix:

$$X^t_{e_{tgt}} = \bigoplus_{v_j^i \in S_{e^t_{tgt}}} [x(v_j^i)]^\top \in R^{d_{enc}\times((k+2)\tau)} \tag{10}$$

Where, $\bigoplus$ represents the concatenation and $(.)^\top$ is transpose operation.

C. Dynamic Graph Transformer Module

In order for the neural network model to draw knowledge from dynamic graphs, it must take into account both the temporal and the spatial structure information. For effective anomaly detection, geographical and temporal information should typically be recorded simultaneously because they are coupled in most circumstances. Using the dynamic graph (Fig. 4), one can observe that at time t, the nodes v_1^t and v_2^t are connected. This is preceded by multiple connections in respective communities in the preceding timestamps, such as $u_1^{(t-1)} - u_2^{(t-1)}$ and $u_1^{(t-2)} - u_2^{(t-2)}$. How can a neural network-based encoder take into account both spatial and temporal information at the same time while designing a dynamic graph encoder? In the works that are now available, using hybrid networks layered by distance based or time dependent modules is a common solution. In these hybrid models, the spatial and temporal modules are used to separately and respectively collect spatial and temporal information. For example, in StrGNN [11], the GCN functions as a space-dependent module, while temporal information is gathered by GRU evaluating the GCN outputs from various timestamps. One drawback of these hybrid frameworks can be considered as they could overlook some data that spans temporal and geographical domains, which would result in an even less-than-ideal outcome. The proposed work utilizes transformer architecture in such a way that only encoder module is fed the spatial, temporal and centrality knowledge of dynamic graphs. In this way, the transformer simultaneously capture all the spatial, centrality and temporal attributes with a single encoder when it gets the input of multiple encoding timestamps. The transformer module and the pooling module make up the dynamic graph transformer. A better representation for the dedicated graph transformer is provided in Fig. 3. The attention mechanism leverages the transformer module to gather the wealth of cross-domain knowledge, and its last layer develops informative node encodings. Finally, subgraph node embedding for each node is then combined by pooling module and an embedding vector is produced to portray the target edge.

1) Transformer Module: Consolidating node encodings within a subgraph node set into node embeddings is the transformer module's goal. Several layers of attention mechanism is used to exchange the data of various nodes in order to achieve this. The single head attention layer can be expressed specifically as follows:

$$H^{(l+1)} = attention(H^{(l)}) \tag{11}$$

$$\hat{h}_i^{l+1} = \bigoplus_{k=1}^{H}\left(\sum_{j\in N} V^{k,l}w_{i,j}^{k,l}h_j^l\right) \tag{12}$$

Where, $H^{(l+1)}$ is the output embedding of the $(l+1)^{th}$ layer, h_j^l is the adjacency value for the j^{th} channel, $w_{i,j}^{k,l}$ is the trainable attention matrix, which is:

$$w_{i,j}^{k,l} = softmax_j\left(\frac{Q^{k,l}h_i^l \cdot K^{k,l}h_j^l}{\sqrt{d_{emb}}}\right) \tag{13}$$

Here, $Q^l, K^l, V^l \in R^{d_{emb}\times((k+2)\tau)}$ are the *query*, *key*, and *value* matrices, respectively for the feature transformation and information exchange. Q^l, K^l, V^l are calculated by:

$$Q^l = H^{l-1} \cdot W_Q^l \tag{14}$$

$$K^l = H^{l-1} \cdot W_K^l \tag{15}$$

$$V^l = H^{l-1} \cdot W_V^l \tag{16}$$

Where $W_Q^l, W_K^l, W_V^l \in R^{d_{emb}\times d_{emb}}$ are learnable parameter matrices of the l^{th} attention layer. The purpose of Q^l and K^l is to sum up the contributions of particular node embeddings whereas V^l is used to set input for new feature space. In the proposed method, the target encoding matrix $X_{e_{tgt}^t}$ acts as input of transformer $H^{(0)}$, and to articulate the dimension, $d = d_{emb} = d_{enc}$. The output node embedding matrix Z is the last attention layer output. In between, the attention outputs h_i^{l+1} made to go through a feed forward neural network preceded and succeeded by unused connections and layer normalization as:

$$h_i^{l+1} = Norm(h_i^l + h_i^{l+1}) \tag{17}$$

$$\hat{h}_i^{l+1} = w_2^l \cdot GeLU(w_1^l h_i^{l+1}) \tag{18}$$

$$\hat{\hat{h}}_i^{l+1} = Norm(h_i^{l+1} + \hat{h}_i^{l+1}) \tag{19}$$

Where, $Norm(.)$ is the layer normalization used, while $GeLU(.)$ [36] is the Gaussian Error Linear Unit. Compared to sigmoids, activations like ReLU, ELU, and PReLU have allowed neural networks to converge more quickly and effectively. Moreover, Dropout multiplies some of the activations by 0 at random to regularize the model. Together, the two aforementioned approaches determine a neuron's output. Nevertheless, the two operate apart from one another. GeLU seeks to integrate them. Mathematically, in a much simplified manner:

$$GeLU(y) = y.P(Y \le y) = y \cdot \Phi(y)$$
$$GeLU(y) \approx 0.5y\left(1 + \tanh\left[\sqrt{2/\pi}(0.04471 \cdot y^3 + y\right]\right) \tag{20}$$

Where $\Phi(y)$ is the cumulative of Gaussian distribution and is generally computed with the error function.

2) Pooling Module: The embeddings of the nodes in subgraph Z are to be transferred over the edge embedding vector $z(e_{tgt}^t)$ via the pooling module. Here, the pooling function is defined as follows:

$$Z_{e_{tgt}^t} = pooling(Z) = \sum_{k=1}^{n_s}\frac{Z_k}{n_s} \tag{21}$$

In here, average pooling operation is achieved, with n_s being the number of nodes in a subgraph and Z_k being the k^{th} row of Z.

D. Binary Cross Entropy based Anomaly Detector

The aim of the proposed work is to determine an anomaly score for all the edges present in the dynamic graph after the edge embedding has been obtained. The framework for anomaly score computation using a fully connected layer based anomaly points detector, is presented. Evidently, the training set in the proposed methodology lacks a ground-truth anomaly sample. A new problem arises in such a scenario: how can an anomaly detector be learned in the absence of an aberrant sample? The proposed work presents a negative sampling strategy to create fake anomalous edges, which is used to train the detector with edges that already exist in the training data.

The major challenge in here is that *How to train such a detector where anomalous data is not provided?*. To address this, pseudo anomalous edges are generated via negative sampling technique and detector is trained with both data as existing and pseudo anomalous altogether. Within the proposed architecture, a basic negative sampling approach is implemented. Same amount of node pair is randomly selected as potential negative pair candidates for each timestamp of a network with m edges. Next, each of these node pairs without connection to current normal edge set are verified across all the timestamps. Up until that node pair is valid, another pair is resampled and every illegal node pair is assessed. Post negative sampling, a centrality enabled spatial-temporal node encoding is present to provide context sampling to get subgraph node set of every negative edge. The negative edge is then embedded by feeding the encoding into the dynamic graph transformer framework.

A fully connected layer is utilized having sigmoid activation function to serve as scoring module for the proposed framework to get anomaly score, and is given by:

$$f(z) = \sigma\left(E(z)wt_s + bias_s\right) \tag{22}$$

Where, $f(z)$ give the anomalous score, $E(x)$ is edge embedding of edge z, $wt_s \in R^{d_{emb}}$ and $bias_s \in R$ are weight and bias parameters respectively.

After that, *Binary Cross Entropy* loss is implemented as:

$$L = -\sum_{i=1}^{m^t} log(1 - f(e_{pos,i})) + log(f(e_{neg,i})) \tag{23}$$

Where, $e_{pos,i}$ represents i^{th} positive edge and $e_{neg,i}$ represents i^{th} negative edge that is negatively sampled, giving out actual loss over prediction.

The model is trained in iterative manner going end-to-end. For every iteration, to prevent over-fitting and bias, different negative edges are sampled. For all the edges, subgraph sampling is followed with the cross-coupled node encoding, transformer model learning and anomaly computation is performed step wise. After that, the parameters are back-propagated and updated under the guidance of loss function.

V. Illustrative System Model

As an example, a social media network is presented (Fig. 3) as the system model, which is one of the model frameworks

Algorithm 1 Proposed Model

Require: Graph $G = \{D_{t=1}^{T}\}$ with node features X,number of contextual nodes sampled k, anomaly labels $Z(e)$, epochs count I, time f each window τ

Ensure: Anomaly scores for nodes in $D_{t=1}^{T}$ Dynamic Graph training data $G = \{D_{t=1}^{T}\}$, epochs count I, sampled node count k, time of each window τ

Initialize temporal embedding x_{temp}
Initialize distance based spatial embedding x_{dist}
Initialize diffusion based spatial embedding x_{diff}
Initialize centrality embedding x_{cent}
Initialize Graph Transformer H^{l}
Initialize Discriminative Anomaly Detector L
for each epoch I **do**
  for each batch in k **do**
    $X_{temp} \leftarrow F_{em_{temp}}(x_{temp})$ {Embed temporal features}
    $X_{dist} \leftarrow F_{em_{dist}}(x_{dist})$ {Embed distance features}
    $X_{diff} \leftarrow F_{em_{diff}}(x_{diff})$ {Embed diffusion features}
    $X_{cent} \leftarrow F_{em_{cent}}(x_{cent})$ {Embed centrality features}
    $X_{e_{tgt}} \leftarrow X_{temp} \oplus X_{dist} \oplus X_{diff} \oplus X_{cent}$ {Combine embeddings}
    $Z_{e_{tgt}} \leftarrow H(X)$ {Transform embeddings with Graph Transformer}
    $S \leftarrow L(Z_{e_{tgt}})$ {Compute anomaly scores with Anomaly Detector}
    Compute loss and update model parameters
  end for
end for
return S =0

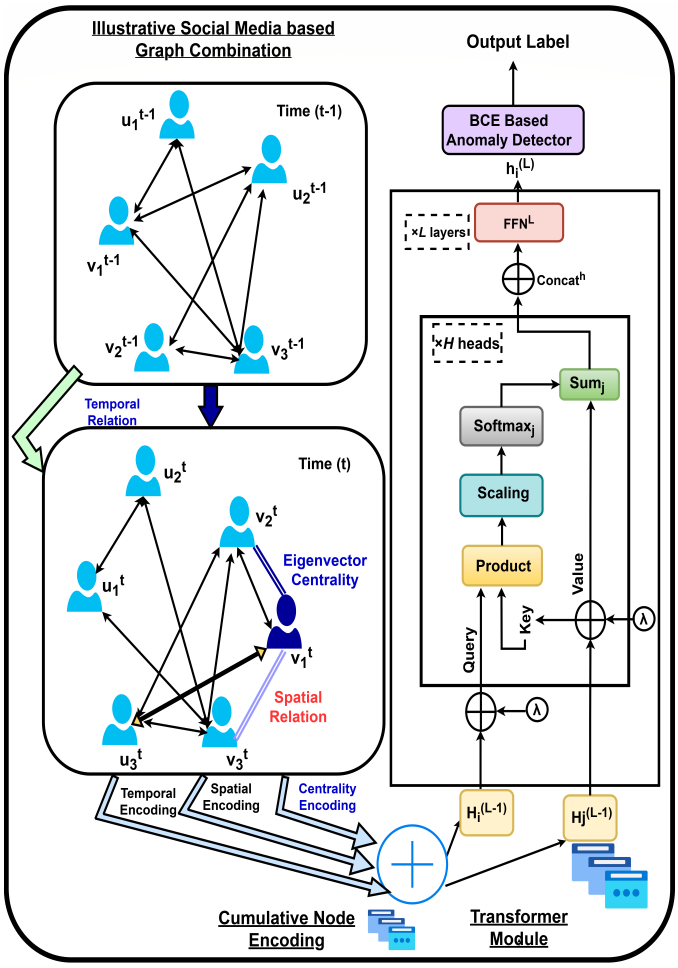


Fig. 3: An Illustration of a Social Media Network Showing Working of Transformer Inside Dynamic Graph Transformer Module

that the proposed methodology is presenting. This illustration considers the UCI Messages dataset [14] environment and is imitated just like a social media platform. The graph dataset is divided into different snapshots and each snapshot serves as a time frame. Here, to give a proper comparison, nodes represent users, edges represent connections between the users. Even if a user has sent a single message to any user, an edge is formed. To represent it dynamically, users add and remove other connections. Other users also join the network and new nodes are getting added. An anomaly may occur, if a user connection is getting established without necessity. With the use of this data, a dynamic network is constructed in which each user is depicted as a node while edges are established by messages that are sent between users. The associations and interactions, altering inside the platform, are captured in this dynamic graph. In order to extract the features from this graph data, various techniques and formulae are implemented that are mentioned in the section IV in detail. Specific time frame τ to consider each snapshot and extract the temporal embedding is set. For spatial embedding, both local and global feature scale is utilized for both distance-based and diffusion-based spatial embedding. In order to effectively extract the centrality embedding feature from the dynamic graph, the eigenvector based centrality scoring scheme is implemented and embedded. All the computations are performed in *Python* using *Sypder* cross-platform Integrated Development Environment (IDE). After efficiently extracting all the features, a cumulative node encoding is developed that is further used to train the proposed transformer neural network.

VI. Theoretical Considerations and Convergence Analysis

The proposed graph-based anomaly detection technique is designed to extract a fixed-size circumstantial node set that ensures cross-coupled temporal, spatial, and centrality-based information. The proposed technique ensures that the circumstantial node set remains bounded in size regardless of the dynamic nature of the graph, making the subsequent transformer-based encoding process computationally stable. The proposed method reiterates the robust convergence properties of transformers, when paired with appropriate encoding and learning-rate schedules. The results across the identified datasets demonstrate consistent convergence of the model during training, as indicated by the steadily decreasing loss curves and performance metrics. A step-by-step mathematical

proof of convergence for the proposed model is presented. The aim is to demonstrate that the parameters $\theta$of the transformer are updated in a manner that reliably decreases the BCE loss function $\text{BCE}(\theta)$, converging to a local minimum. This proof relies on properties such as Lipschitz continuity and the smoothness of the gradients.

A. Model Configuration and BCE Loss Function

The BCE loss for binary classification tasks is defined as follows:

$$\text{BCE}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(\hat{y}_i(\theta)) + (1-y_i)\log(1-\hat{y}_i(\theta))\right] \tag{24}$$

where y_i are labels, and $\hat{y}_i(\theta)$ are the predicted probabilities, modeled by the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$.

B. Gradient Descent Update Rule

The parameters are updated using gradient descent as follows:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_\theta \text{BCE}(\theta^{(t)}) \tag{25}$$

where η is the learning rate and $\nabla_\theta \text{BCE}(\theta^{(t)})$ is the gradient of the BCE loss at iteration t.

C. Lipschitz Continuity and Gradient Smoothness

Assume the gradient of the BCE loss $\nabla_\theta \text{BCE}(\theta)$ is Lipschitz continuous, i.e., there exists a constant $L > 0$ such that:

$$\|\nabla_\theta \text{BCE}(\theta_1) - \nabla_\theta \text{BCE}(\theta_2)\| \leq L\|\theta_1 - \theta_2\| \tag{26}$$

for all θ_1, θ_2.

D. Convergence Proof

Using Taylor's expansion and the mean value theorem, we approximate the BCE loss around $\theta^{(t)}$ as:

$$\text{BCE}(\theta^{(t+1)}) \approx \text{BCE}(\theta^{(t)}) + \nabla_\theta \text{BCE}(\theta^{(t)})^T(\theta^{(t+1)} - \theta^{(t)}) \tag{27}$$

$$+ \frac{1}{2}(\theta^{(t+1)} - \theta^{(t)})^T H(\theta^{(t+1)} - \theta^{(t)}) \tag{28}$$

By choosing a sufficiently small learning rate η, specifically $\eta < \frac{2}{L}$, we ensure that:

$$\text{BCE}(\theta^{(t+1)}) < \text{BCE}(\theta^{(t)}) \tag{29}$$

proving that the loss function decreases with each iteration, leading to convergence of the model parameters θ to a local minimum under the assumption of gradient smoothness and proper learning rate selection.

This proof under reasonable mathematical assumptions demonstrates that our model's training process, characterized by the gradient descent optimization of the BCE loss, converges to a local minimum, ensuring effective learning and stability of the anomaly detection model in dynamic graphs.

VII. Datasets and Experiments

A. Dataset Description and Pre-processing

The proposed framework is evaluated with the help of six benchmark datasets from the real world. Every dataset has time-stamped annotations on its edges. During the pre-processing stage, the edge stream's repeated edges are eliminated. The method described in [8] is applied to introduce anomalous edges into each dataset because the original datasets missed a ground-truth anomalous edge. The training data is completely clean. $X_a \times m_t$ pairs of different nodes are associated arbitrarily for each snapshot D^t and is associated as anomalous nodes in the test data set. Here, m_t is the actual count of edges in the graph, and X_a is the anomaly amount, which shows the proportion of anomalous edges corresponding to each time-frame. We train UCI Messages [14], Bitcoin-Alpha [15], and Digg [16] datasets with 100 epochs and Enron Email [17], Epinions-Trust [18] and AS-Topology [19], datasets for 200 epochs. The snapshot size is set to be $1,000$ for UCI Messages, Bitcoin, and Digg, 2000 for Enron and Bitcoin-Alpha, and 6000 for Epinions-Trust and AS-Topology, respectively. Table I summarizes the important features and statistics of the used datasets.

TABLE I: Important Features and Dataset Statistics

Features → / Dataset ↓	No. of Nodes	No. of Edges	Avg. Degree	Gini Coefficient	Mean Distance
UCI Messages [14]	1899	13838	14.57	0.75	3.1
Bitcoin-Alpha [15]	3777	24173	12.80	0.70	3.6
Digg [16]	30360	85155	5.7	0.63	4.6
Enron Email [17]	87000	1100000	26	0.75	4.9
Epinions-Trust [18]	131828	841372	12.76	0.9423	4.10
AS-Topology [19]	34,761	0.72	9.86	0.80	3.7

B. Computational Resource Requirements

The proposed method was implemented and evaluated on a high-end system, equipped with an NVIDIA RTX 3090 GPU (24GB VRAM), an AMD Ryzen 9 5950X processor (16 cores, 32 threads), and 64 GB of RAM, running Ubuntu 20.04 LTS with PyTorch 2.0 and CUDA 11.8. This setup enabled rapid experimentation, with training times of approximately 2–5 minutes per epoch depending on the dataset and efficient inference times within $2-3$ seconds for test samples.

C. Baseline Models and Validation Parameters

The performance of the proposed framework is validated on *eight* parameters and also compared with *eight* forefront baseline models. The details of the same are presented in Table II. With the aim of a fair evaluation, every dataset is split into two subsets: the test set comprises the 50% of timestamps, and the training set consists of the remaining 50%. The test introduces the anomalous data at three distinct anomaly rates (pA): 1%, 5%, and 10%. Based on the learned node embeddings, the clustering technique oriented anomaly detector [8] is used for the graph embedding technique to find anomalies.

TABLE II: Baseline Models and Validation Parameters

S.No.	Method Name ↓	Parameters ↓
1	NetWalk [8]	Sensitivity Analysis
2	StrGNN [11]	AUC-ROC with Varying Anomaly Rate
3	AddGraph [22]	AUC-ROC with Varying Epochs
4	TADDY [24]	Computational Cost Analysis
5	Node2vec [26]	Accuracy with Varying Epochs
6	DeepWalk [25]	Precision with Varying Epochs
7	Spectral Clustering [28]	Scalability
8	RustGraph [35]	Ablation Analysis

VIII. Hyper-parameter Tuning and Sensitivity Analysis

To investigate the impact of hyper-parameter tuning on the proposed method, a comprehensive analysis is performed. The analysis considers a *Small Scale Dataset* i.e Bitcoin-Alpha [15] and considers *Snapshot Size* and *Training Ratio*, as the two hyper-parameters. Additionally, a *Large Scale Dataset* i.e. Epinions-Trust [18], is considered for validation on the same hyper-parameters. Experiments are performed with 10% anomaly rate while the remaining parameters remain as default.

A. Analysis on Small Scale Dataset

1) Sensitivity Analysis based on Snapshot Size: The snapshot sizes impact the model as lower snapshot size leads to low information retrieval and model underfitting, whereas very large snapshot sizes often leads to over-fitting. The results on the identified dataset is presented in Table III for the proposed and two of the most recent methods. A careful observation reveals that at smaller screenshot sizes, limited number of nodes result in limited structural information and thereby impacting the models performance. Interestingly, with larger screenshots, the temporal information is significantly improved resulting in improved models performance. The proposed method is able to outperform the compared methods with significant margin where the highest performance is marked at 0.93 and 0.90 for respective edges of the evaluation sizes. The results reiterate the importance of cross-coupled temporal, spatial, and centrality-based information which the proposed method considers for improved outcome.

TABLE III: Small Scale Dataset Sensitivity Analysis - Snapshot Size

Method ↓	Bitcoin-Alpha [15]				
Snapshot Sizes →	100	500	1000	2000	5000
TADDY [24]	0.7842	0.9240	0.9427	0.9423	0.8994
RustGraph [35]	0.9353	0.9412	0.9376	0.9207	0.8975
Proposed Model	0.9383	0.9441	0.9365	0.9212	0.9017

2) Sensitivity Analysis Based on Training Ratios: With an anomaly score of 10% and for a set of varying training ratios (20%, 30%, 40%, 50% or 60%), a thorough comparative analysis is made between the proposed method, TADDY [24] and RustGraph [35]. Fig. 4 shows the box-plot analysis of the results obtained for the identified methods. It is observed that with the increase in training ratio, a stepp proliferation is seen in the AUC score, validating the model's ability to analyze and find the best fit. The variance in the proposed model decreases with increase in training ratio, whereas no such shift is observed in TADDY [24] and RustGraph [35], proving the dexterity of the proposed method to capture the maximum information to achieve better results.

B. Analysis on Large Scale Dataset

1) Sensitivity Analysis based on Snapshot Size: The Table IV presents a comprehensive performance analysis of TADDY, RustGraph, and the Proposed Model on the Epinions-Trust dataset. As evident from the results, the proposed method is able to consistently outperform the other methods, with its performance peaking at the largest snapshot size of 5000, demonstrating scalability and robustness. The models high performance even with large datasets is the outcome of it's graph-based diffusion technique, which samples fixed-size, cross-coupled circumstantial node sets for target edges, and its centrality-enabled spatial-temporal node encoding. The results validate the claim that the dynamic graph-based transformer is able to input a rich representation, capturing extensive spatial-temporal attributes within a single encoder. It is empirical to note that hyperparameter tuning plays a pivotal role in optimizing these encodings, influencing how effectively the transformer captures and integrates spatial-temporal dependencies.

TABLE IV: Large Scale Dataset Sensitivity Analysis - Snapshot Size

Method ↓	Epinions-Trust [18]				
Snapshot Sizes →	100	500	1000	2000	5000
TADDY [24]	0.8437	0.9387	0.9553	0.8902	0.9184
RustGraph [35]	0.8809	0.9399	0.8974	0.9042	0.9201
Proposed Model	0.9147	0.9492	0.9525	0.9591	0.9710

2) Sensitivity Analysis Based on Training Ratios: The Fig. 5 shows the models performance with respect to TADDY [24] and RustGraph [35], on the Epinions-Trust dataset across varying training ratios. The models encoding strategy allows a rich input representation making the dynamic graph-based transformer to simultaneously capture spatial and temporal dependencies in a single encoder. Furthermore, with hyperparameter tuning, the models ability is optimized to generalize across different training ratios, ensuring efficient learning even with limited training data. Results show that the proposed model consistently outperforms the other methods, with its performance improving as the training ratio increases, peaking at the highest ratio of 90%. The proposed model's robust encoding, enhanced by effective tuning, ensures superior adaptability and scalability, making it particularly well-suited for datasets with varying training ratios like Epinions-Trust.

IX. Performance Validation

The performance analysis of the proposed method against the identified baseline techniques, is presented on a set of evaluation metrics. Specifically, *AUC-ROC*, *Accuracy*, *Average Precision*, *Scalability*, *Computation Cost*, *Memory Usage* and *Ablation Analysis*, are used as the metrics for performance validation. All the methods are implemented and validated on the identified datasets for the predefined parameters.

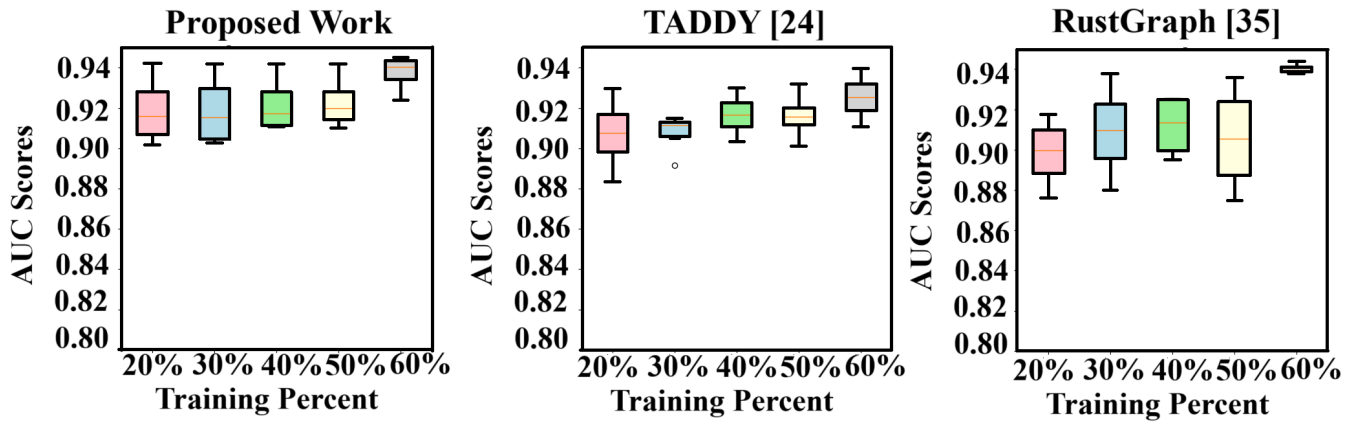


Fig. 4: Small Scale Dataset Sensitivity Analysis - Training Ratio

A. AUC-ROC with Varying Anomaly Rate

The performance comparison of the proposed method in terms of AUC-ROC value with varying anomaly rates is presented in Table V and Fig. 6. The values are presented for all the six identified datasets i.e Digg [16], Bitcoin-Alpha [15], UCI Messages [14], Enron Email [17], Epinions-Trust [18] and AS-Topology [19], for all the baseline methods on three different anomaly rates of 1%, 5% and 10%. As evident from the AUC measure on the anomaly scores in the Table V and Fig. 6, the proposed method outperforms the state-of-the-art techniques with a clear margin. The best performance margin of the proposed method is reported at 0.9436 (maximum) on the Bitcoin-Alpha [15] dataset (small scale) with an anomaly score of 10% and for Epinions-Trust [18] dataset (large scale) where the value peaks at 0.9785 at 10% anomaly rate. The performance validates the claim of the proposed method of being scale-invariant. Overall, the best results are observed with Bitcoin-Alpha and Epinions-Trust, with most stable results for each type of anomaly composition with AUC-ROC score averaging to be 94.36% and 0.9785% respectively. The results prove the models performance with varying anomaly rates on other datasets as well where the results are marked for UCI Messages [14] dataset at an average AUC-ROC score of 86.76%. For a smaller anomaly rate, i.e. 1%, UCI Messages dataset (small scale) gives the best improvement of 0.5% and Enron Email [17] shows an average improvement of 8.7%, when compared with the state-of-the-art methods. For the largest dataset under consideration i.e. Epinions-Trust [18], the proposed method is able to achieve an average AUC-ROC score of 95.78% which is significantly better than most the compared methods.

B. AUC-ROC with Varying Epochs

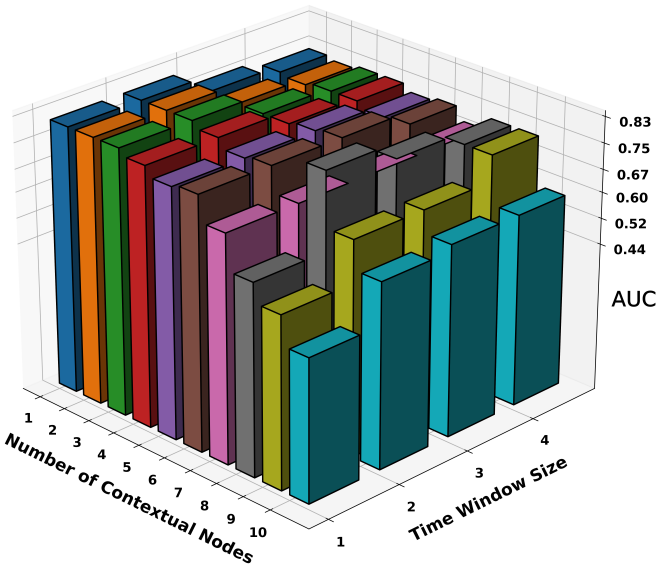
The Fig. 7 presents the AUC-ROC performance of the proposed approach with varying epochs. The dataset used for the validation is UCI Messages [14] and the anomaly rate is considered to be 1%. The proposed method is able to counter over-fitting and the results can be verified from the AUC-ROC and Loss performance with varying epochs. The presented results are obtained by running the model through various embedding techniques, such as *time embedding*, *combining time embedding with diffusion based spatial embedding*, *time with distance based spatial embedding*, *time embedding with centrality embedding* (referred as *T*, *T + Dif*, *T + Dif*, *T + Dis*, *T + C*, respectively in the graph) and a combination of all four. The findings are consistent with the claims for the proposed method having superior AUC-ROC score and also has minimum loss rate.

C. Accuracy with Varying Epochs

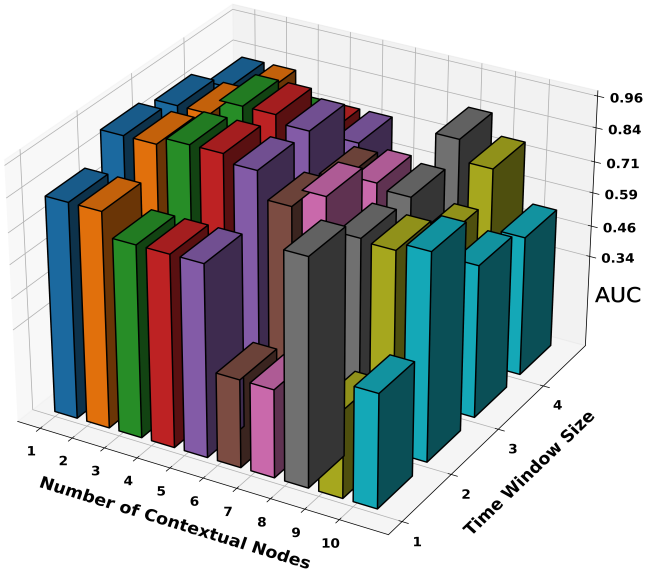
The Fig. 8 presents the accuracy performance of the proposed approach with varying epochs. The dataset used for the validation is Bitcoin-Alpha [15] and the anomaly rate is considered to be 1%. The presented results are obtained by running the model through various embedding techniques, such as *time embedding*, *combining time embedding with diffusion based spatial embedding*, *time with distance based spatial embedding*, *time embedding with centrality embedding* (referred as *T*, *T + Dif*, *T + Dif*, *T + Dis*, *T + C*, respectively in the graph) and a combination of all four. The results prove that the proposed method is able to achieve accuracy of as high as 96% which is a fairly high value and validates the performance claims made for the proposed method.

D. Precision with Varying Epochs

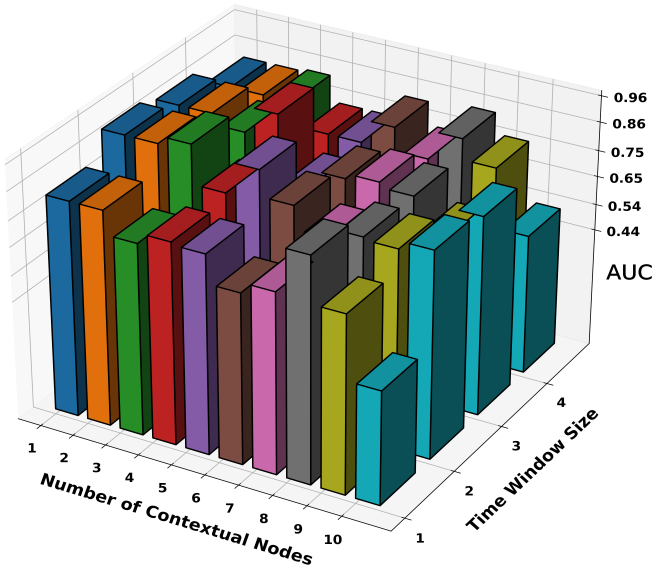
The Fig. 9 presents the average precision performance of the proposed approach with varying epochs. The dataset used for the validation is Bitcoin-Alpha [15] and the anomaly rate is considered to be 1%. The presented results are obtained by running the model through various embedding techniques, such as *time embedding*, *combining time embedding with diffusion based spatial embedding*, *time with distance based spatial embedding*, *time embedding with centrality embedding* (referred as *T*, *T + Dif*, *T + Dif*, *T + Dis*, *T + C*, respectively in the graph) and a combination of all four. A careful observation of the results show a steady increase at start, which reflects improvement while training and a stable value above 0.5 clearly depicts that the model has converged with high precision without over-fitting.



(a) Performance - Proposed Work



(b) Performance - TADDY [24]



(c) Performance - RustGraph [35]

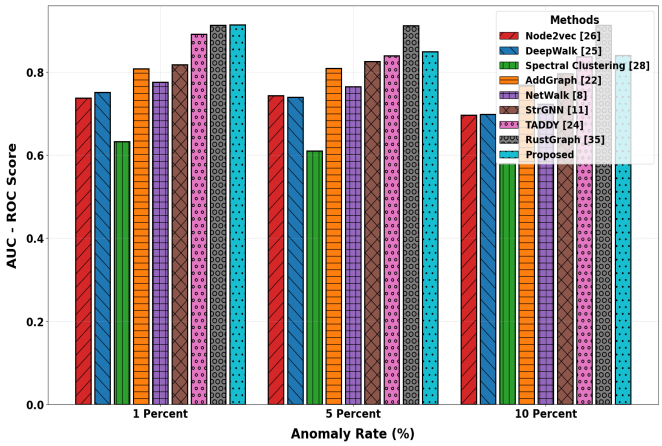
Fig. 5: Large Scale Dataset Sensitivity Analysis - Training Ratio

E. Scalability

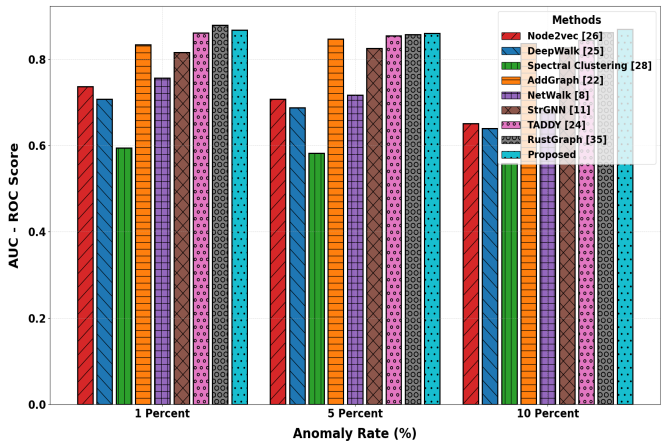
The performance reprot of all the methods, presented in Table V, clearly shows that the proposed model is highly scalable and is able to give steady performance even for large scale datasets. The impact of subgraph sampling and efficient node encoding strategy allows each node, within the sampled subgraph, to compute various embeddings i.e. centrality-based, temporal, and spatial. This impacts in proliferation in the models performance even when the anomaly percent is high and the size of the network grows to more than 130000 nodes. Another important component is the *Dynamic Graph Transformer Module*, which incorporates a self-attention mechanism and allows pairwise interactions among the nodes and node feature transformations, leading to improved performance and outperforming the state-of-the-art approaches. It is pertinent to note that while the performance of the proposed method is very good, at small scale datasets such as UCI Messages [14] and Bitcoin-Alpha [15], the models performance furhther improves with large scale datasets such as Enron Email [17] and Epinions-Trust [18].

F. Computational Cost Analysis

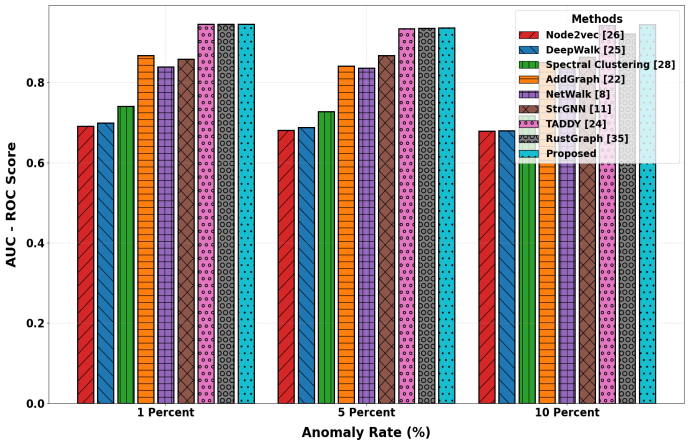
To present a comprehensive analysis of the computational cost, we analyze the computational complexity of each component of the proposed anomaly detection framework. For *Subgraph Sampling*, the complexity primarily stems from the number of edges e and the neighborhood size k, expressed as $O(e \cdot k)$. In *Node Encoding*, each node within the sampled subgraph is processed to compute various embeddings such as centrality-based, temporal, and spatial, resulting in a com-



(a) Performance Comparison - UCI Dataset



(b) Performance Comparison - Digg Dataset



(c) Performance Comparison - Bitcoin Dataset

Fig. 6: Anomaly Detection Performance Comparison: AUC-ROC Score

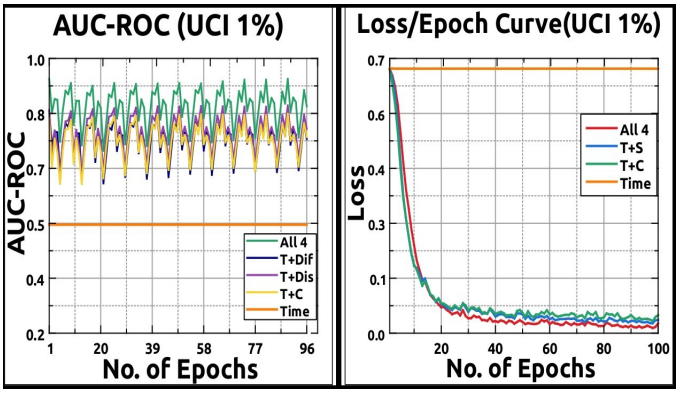


Fig. 7: Anomaly Detection Performance Comparison with Varying Epochs: AUC-ROC and Loss Curve

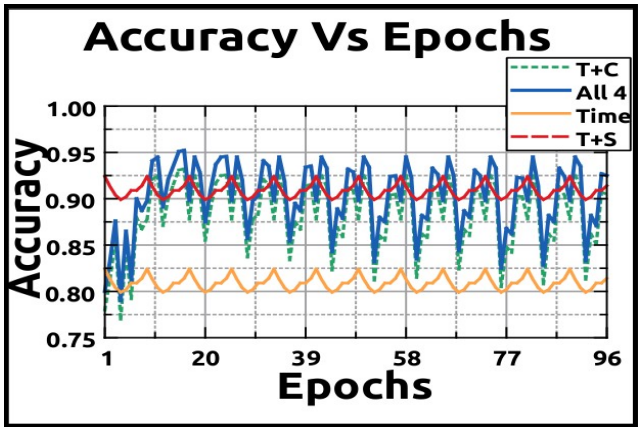


Fig. 8: Anomaly Detection Performance Comparison with Varying Epochs: Accuracy.

plexity of $O(n \cdot c)$ where n is the number of nodes and c is the constant time for encoding each node. The core of the model, the *Dynamic Graph Transformer Module*, incorporates a self-attention mechanism with a complexity of $O(n^2 \cdot d)$ due to pairwise interactions among the nodes, where d is the dimension of embeddings. For *Anomaly Detection*, the complexity is $O(e \cdot d)$, mainly involving a sigmoid activation computed over edge features. Summarizing, the overall computational complexity of the system, which primarily involves processing the interactions within subgraphs and node feature transformations, is dominated by the transformer module and is expressed as $O(n^2 \cdot d + e \cdot d)$, showcasing a significant computational demand especially for large-scale graphs.

TABLE V: Anomaly Detection Performance Comparison: AUC-ROC Measure

Method ↓	UCI Messages [14]			Bitcoin-Alpha [15]			Digg [16]		
Anomaly Percent →	1%	5%	10%	1%	5%	10%	1%	5%	10%
NetWalk [8]	0.7758	0.7647	0.7226	0.8385	0.8357	0.8350	0.7562	0.7175	0.6836
StrGNN [11]	0.8179	0.8252	0.7959	0.8667	0.8627	0.8162	0.8254	0.8271	0.8574
AddGraph [22]	0.8083	0.8090	0.7688	0.8665	0.8403	0.8498	0.8341	0.8470	0.8369
TADDY [24]	0.8912	0.8398	0.8370	0.9451	0.9341	0.9423	0.8614	0.8545	0.8440
Node2vec [26]	0.7371	0.7433	0.6960	0.6910	0.6802	0.6785	0.7364	0.7081	0.6508
DeepWalk [25]	0.7512	0.7390	0.6978	0.6985	0.6874	0.6793	0.7080	0.6881	0.6396
Spectral Clustering [28]	0.6324	0.6103	0.5795	0.7401	0.7275	0.7167	0.5949	0.5823	0.5591
RustGraph [35]	0.9128	0.9117	0.9124	0.9447	0.9348	0.9207	0.8795	0.8577	0.8624
Proposed Work	0.9141	0.8487	0.8402	0.9453	0.9357	0.9436	0.8680	0.8601	0.8695

Method ↓	Enron Email [17]			Epinions-Trust [18]			AS-Topology [19]		
Anomaly Percent →	1%	5%	10%	1%	5%	10%	1%	5%	10%
NetWalk [8]	0.7562	0.7175	0.6836	0.8385	0.8357	0.8350	0.7758	0.7647	0.7226
StrGNN [11]	0.8162	0.8254	0.8271	0.8574	0.8667	0.8627	0.8179	0.8252	0.7959
AddGraph [22]	0.8341	0.8470	0.8369	0.8665	0.8403	0.8498	0.8083	0.8090	0.7688
TADDY [24]	0.8614	0.8545	0.8440	0.9451	0.9341	0.9423	0.8912	0.8398	0.8370
Node2vec [26]	0.7364	0.7081	0.6508	0.6910	0.6802	0.6785	0.7371	0.7433	0.6960
DeepWalk [25]	0.7080	0.6881	0.6396	0.6985	0.6874	0.6793	0.7512	0.7390	0.6978
Spectral Clustering [28]	0.5949	0.5823	0.5591	0.7401	0.7275	0.7167	0.6324	0.6103	0.5795
RustGraph [35]	0.8795	0.8577	0.8624	0.9447	0.9348	0.9207	0.9128	0.9117	0.9124
Proposed Work	0.8680	0.8601	0.8695	0.9453	0.9357	0.9436	0.9141	0.8487	0.8402

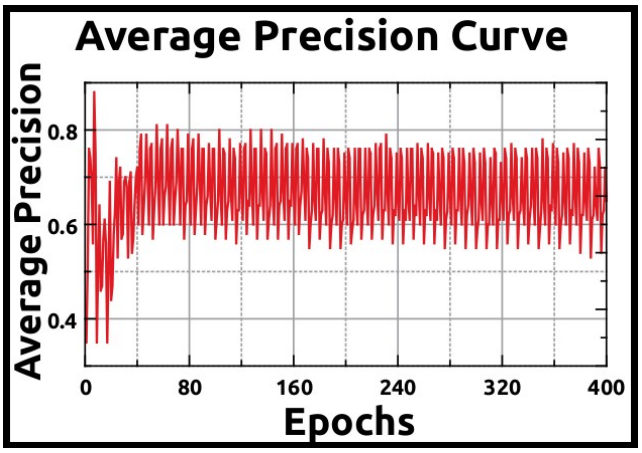


Fig. 9: Anomaly Detection Performance Comparison with Varying Epochs: Average Precision

G. Ablation Analysis

The ablation investigation of the proposed framework is presented in order to examine how each element of the spatial, temporal and centrality node encoding contributes to it's overall performance. The dataset used for the validation are UCI Messages [14], Bitcoin-Alpha [15], Enron Email [17] and Epinions-Trust [18] and the anomaly rate is considered to be 1%. Based on the findings, the following observations can be drawn:

1) The temporal encoding is observed to be the most important for the node encoding as this alone impacts about 50% of the AUC-ROC score.
2) In identifying anomalies, all the three; temporal encoding, centrality encoding as well as spatial encoding, serve a slightly smaller role.
3) Eliminating one of them would, in most situations, result in a modest decline in performance.
4) Combining all forms of encoding typically yields the most significant AUC-ROC values.
5) Considering each encoding one by one, is of very less significance as it contains maximum loss.
6) Considering the loss curve to be diverging and minimizing the loss, it can be inferred that the model is able to avoid over-fitting. Over-fitting, in this case, refers to the phenomenon where the model is extensively capturing noise or specific patterns from the training data, hindering its ability to generalize to unseen data or alter the shift in the graph structure along with time. This leads to poor model performance after training. The proposed model deals with over-fitting by implementing effective regularization steps in the dynamic graph transformer and early stopping criteria in loss function.

X. Conclusion and Future Directions

This work presents an innovative transformer-based framework for dynamic graph anomaly detection. The proposed work addresses the challenges of capturing spatial, temporal, and centrality-based cross-coupled information. The designed end-to-end anomaly detection system consists of four components: subgraph sampling based on edges, node encoding based on multiple embedding strategies, dynamic graph transformer and pooling module, and binary cross entropy based anomaly detector. The proposed methodology leverages a single transformer model that captures the cross-coupled spatial,

centrality and temporal details within dynamic networks. This allows effective representation of the functions of nodes in a developing graph space through an informative and detailed node encoding. The proposed method achieves superior performance across six diverse datasets, demonstrating robustness, scalability and accuracy. By outperforming existing techniques in key metrics such as AUC-ROC, accuracy, and precision, the approach highlights its scalability and effectiveness in both small and large-scale scenarios.

We continue to work towards extending the framework to address challenges such as handling sparsity in dynamic graphs and adapting the model to multi-modal graph data with heterogeneous node and edge types. Working towards the model's practicality for real-time and large-scale scenarios, healthcare systems would be considered. Advanced metrics such as memory usage, interpretability, robustness to adversarial attacks, and anomaly explainability may be used to provide deeper insights into the model's performance. An interesting approach to enhance the framework would be to capture long-term temporal dependencies, which could improve its effectiveness in detecting complex patterns in evolving networks.

REFERENCES

[1] S. Ji, S. Pan, E. Cambria, P. Marttinen, and S. Y. Philip, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE transactions on neural networks and learning systems*, vol. 33, no. 2, pp. 494–514, 2021.

[2] D. Jin, Z. Yu, P. Jiao, S. Pan, D. He, J. Wu, S. Y. Philip, and W. Zhang, "A survey of community detection approaches: From statistical modeling to deep learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 2, pp. 1149–1170, 2021.

[3] F. Xia, K. Sun, S. Yu, A. Aziz, L. Wan, S. Pan, and H. Liu, "Graph learning: A survey," *IEEE Transactions on Artificial Intelligence*, vol. 2, no. 2, pp. 109–127, 2021.

[4] G. Yi and S. Wu, "Graphical visual analysis of consumer electronics public comment information mining under knowledge graph," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 2917–2924, 2024.

[5] S. He, G. Li, T. Yi, O. Alfarraj, A. Tolba, A. Kumar Sangaiah, and R. Simon Sherratt, "Graph structure learning-based multivariate time series anomaly detection in internet of things for human-centric consumer applications," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 3, pp. 5419–5431, 2024.

[6] X. Pan, Q. Deng, Y. Jiao, and Z. Chen, "A variational graph autoencoder aided canonical correlation analysis based online abnormal patterns detection method for buildings HVAC systems," *IEEE Transactions on Consumer Electronics*, pp. 1–1, 2024, early Access. DOI: 10.1109/TCE.2024.3478310.

[7] G. ALMahadin, Y. Aoudni, M. Shabaz, A. V. Agrawal, G. Yasmin, E. S. Alomari, H. M. R. Al-Khafaji, D. Dansana, and R. R. Maaliw, "Vanet network traffic anomaly detection using gru-based deep learning model," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 4548–4555, 2023.

[8] W. Yu, W. Cheng, C. C. Aggarwal, K. Zhang, H. Chen, and W. Wang, "Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 2672–2681.

[9] L. Zheng, Z. Li, J. Li, Z. Li, and J. Gao, "Addgraph: Anomaly detection in dynamic graph using attention-based temporal gcn." in *IJCAI*, vol. 3, 2019, p. 7.

[10] W. Yu, W. Cheng, C. C. Aggarwal, K. Zhang, H. Chen, and W. Wang, "Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 2672–2681.

[11] L. Cai, Z. Chen, C. Luo, J. Gui, J. Ni, D. Li, and H. Chen, "Structural temporal graph neural networks for anomaly detection in dynamic graphs," in *Proceedings of the 30th ACM international conference on Information & Knowledge Management*, 2021, pp. 3747–3756.

[12] D. Javeed, M. S. Saeed, I. Ahmad, P. Kumar, A. Jolfaei, and M. Tahir, "An intelligent intrusion detection system for smart consumer electronics network," *IEEE Transactions on Consumer Electronics*, vol. 69, no. 4, pp. 906–913, 2023.

[13] L. Liu, W. Hamilton, G. Long, J. Jiang, and H. Larochelle, "A universal representation transformer layer for few-shot image classification," *arXiv preprint arXiv:2006.11702*, 2020.

[14] Stanford Network Analysis Project (SNAP), "Uci online social network (messages) dataset," https://snap.stanford.edu/data/CollegeMsg.html, 2014.

[15] Stanford Network Analysis Project, "Bitcoin alpha trust network dataset," https://snap.stanford.edu/data/soc-sign-bitcoin-alpha.html, 2014.

[16] KONECT Project, "Digg reply network dataset," http://konect.cc/networks/munmun_digg_reply/, 2009.

[17] CALO Project, "Enron email dataset," https://www.cs.cmu.edu/~enron/, 2015.

[18] Jure Leskovec and Julian McAuley, "Epinions social network dataset," https://snap.stanford.edu/data/soc-sign-epinions.html, 2014.

[19] KONECT Project, "Internet as-level topology dataset," http://konect.cc/networks/topology/, 2013.

[20] E. Manzoor, S. M. Milajerdi, and L. Akoglu, "Fast memory-efficient anomaly detection in streaming heterogeneous graphs," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1035–1044.

[21] D. Eswaran, C. Faloutsos, S. Guha, and N. Mishra, "Spotlight: Detecting anomalies in streaming graphs," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1378–1386.

[22] L. Zheng, Z. Li, J. Li, Z. Li, and J. Gao, "Addgraph: Anomaly detection in dynamic graph using attention-based temporal gcn." in *IJCAI*, vol. 3, 2019, p. 7.

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

[24] Y. Liu, S. Pan, Y. G. Wang, F. Xiong, L. Wang, Q. Chen, and V. C. Lee, "Anomaly detection in dynamic graphs via transformer," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 12, pp. 12 081–12 094, 2021.

[25] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701–710.

[26] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[27] B. Xu, J. Wang, Z. Zhao, H. Lin, and F. Xia, "Unsupervised anomaly detection on attributed networks with graph contrastive learning for consumer electronics security," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 4062–4072, 2024.

[28] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, pp. 395–416, 2007.

[29] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, 2020, pp. 38–45.

[30] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[31] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim, "Graph transformer networks," *Advances in neural information processing systems*, vol. 32, 2019.

[32] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "Heterogeneous graph transformer," in *Proceedings of the web conference 2020*, 2020, pp. 2704–2710.

[33] Y. Rong, Y. Bian, T. Xu, W. Xie, Y. Wei, W. Huang, and J. Huang, "Self-supervised graph transformer on large-scale molecular data," *Advances in Neural Information Processing Systems*, vol. 33, pp. 12 559–12 571, 2020.

[34] J. Zhang, H. Zhang, C. Xia, and L. Sun, "Graph-bert: Only attention is needed for learning graph representations," *arXiv preprint arXiv:2001.05140*, 2020.
[35] J. Guo, S. Tang, J. Li, K. Pan, and L. Wu, "Rustgraph: Robust anomaly detection in dynamic graphs by jointly learning structural-temporal dependency," *IEEE Transactions on Knowledge and Data Engineering*, 2023.
[36] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

Vishal Krishna Singh received his bachelor's degree in Information Technology, in 2010, the master's degree in Computer Technology and Application, in 2013, and PhD degree in Information Technology from Indian Institute of Information Technology, Allahabad, India in 2018. He is currently working as a Lecturer and is associated with the Networks and Communications Research Group at School of Computer Science and Electronics Engineering, University of Essex, Colchester, U.K. His research interests include Internet of Things, Wireless Sensor Networks, In-Network Inference, Machine Learning and Data Analytics.

Niharika Anand received PhD from the Indian Institute of Information Technology, Allahabad, India. She is currently working as an Assistant Professor in the Department of Information Technology at the Indian Institute of Information Technology, Lucknow, India. Her research areas include Machine Learning, Deep Learning, Federated Learning, Cloud Computing, Internet of Things, Cyber Forensics, 3-D Wireless Sensor Networks, Wireless Sensor Network Localization, and WSN Topology Control and Maintenance.

Amrit Pal received the B.Tech. degree from Kurukshetra University, Kurukshetra, India in 2011, the M.Tech. degree from National Institute of Technical Teachers' Training and Research, Bhopal, India in 2014, and the Ph.D. degree from Indian Institute of Information Technology, Allahabad, India in 2020. He worked as an Assistant Professor at the Centre for Advanced Studies, AKTU, Lucknow, India. He is currently working as an Assistant Professor at the Vellore Institute of Technology, Chennai, India. His research interests include big data analytics, cloud computing, machine learning, and internet of things.

Abishi Chowdhury received the B.E. degree from University Institute of Technology, West Bengal, India in 2011, the M.Tech. degree from National Institute of Technical Teachers' Training and Research, Bhopal, Madhya Pradesh, India in 2014, and the Ph.D. degree from Visvesvaraya National Institute of Technology, Nagpur, India in 2020. She is currently working as an Assistant Professor at Vellore Institute of Technology, Chennai, India. Her research interests include cloud computing, cloud resource scheduling, machine learning, and Internet of Things.

Arjun Srivastava is currently working as a Software Engineer at Dell Technologies, India. He obtained his masters degree from the Indian Institute of Information Technology (IIIT) Lucknow, India, in 2024. His primary research interests are in Machine Learning, Smart Transportation, Graph Networks, Computer Vision, Wireless Communication, and Internet of Things (IoT).