

DyGCL: Dynamic Graph Contrastive Learning For Event Prediction

Muhammed Ifte Khairul Islam
Department of Computer Science
Georgia State University
Atlanta, GA, USA
mislam29@student.gsu.edu

Khaled Mohammed Saifuddin
Department of Computer Science
Georgia State University
Atlanta, GA, USA
ksaifuddin1@student.gsu.edu

Tanvir Hossain
Department of Computer Science
Georgia State University
Atlanta, GA, USA
thossain5@student.gsu.edu

Esra Akbas
Department of Computer Science
Georgia State University
Atlanta, GA, USA
eakbas1@gsu.edu

Abstract—Predicting events such as political protests, flu epidemics and criminal activities is crucial to proactively taking necessary measures and implementing required responses to address emerging challenges. Capturing contextual information from textual data for event forecasting poses significant challenges due to the intricate structure of the documents and the evolving nature of events. Recently, dynamic Graph Neural Networks (GNNs) have been introduced to capture the dynamic patterns of input text graphs. However, these models only utilize node-level representation, causing the loss of the global information from graph-level representation. On the other hand, both node-level and graph-level representations are essential for effective event prediction as node-level representation gives insight into the local structure, and the graph-level representation provides an understanding of the global structure of the temporal graph. To address these challenges, in this paper, we propose a Dynamic Graph Contrastive Learning (DyGCL) method for event prediction. Our model DyGCL employs a local view encoder to learn the evolving node representations, which effectively captures the local dynamic structure of input graphs. Additionally, it harnesses a global view encoder to perceive the hierarchical dynamic graph representation of the input graphs. Then we update the graph representations from both encoders using contrastive learning. In the final stage, DyGCL combines both representations using an attention mechanism, and optimizes its capability to predict future events. Our extensive experiment demonstrates that our proposed method outperforms the baseline methods for event prediction on six real-world datasets.

Index Terms—Event Prediction, Dynamic Graph Neural Networks, Dynamic Graph, Graph Pooling

I. INTRODUCTION

The course of our daily lives is significantly shaped by a variety of human events, including political protests [1], flu epidemics [2], [3], and criminal activities [4]. The proliferation of numerous media outlets has enabled us to apply data-driven machine-learning strategies for predicting future events. This advancement significantly assists local authorities and organizations in making informed decisions and implementing appropriate responses. Recently, deep-learning models have

been utilized for event prediction, where features from the input data are autonomously learned. Among them, Recurrent Neural Networks (RNNs) specifically treat event data as a time series, learning from the historical context from previous time points and using this knowledge to predict future events. However, RNNs primarily concentrate on the semantic information within the input data without considering the temporal structural dependencies between entities related to the event. To capture the dependency between entities, summarized time-series data are represented as dynamic graphs that have changes in interaction and attributes of nodes over time. While small changes are expected over time, major changes in the structure and attributes of the graph can give important information about the rise of events.

To learn semantic structural information in text by composing graphs, dynamic GNNs have been developed as extensions to standard GNNs to capture the dynamic patterns within dynamic graphs and used for event prediction [5], [6]. The existing dynamic GNNs for event prediction typically learn the node-level representation of each timestamp’s graph based on prior timestamp graphs. More specifically they learn static snapshot node embeddings via a GNNs model that induces the local neighborhood properties to the node representations and their evolution is captured by a recurrent neural network architecture. The final node representations are then employed to predict future events. Although these models show promising results, they primarily focus on node-level representation and overlook the graph-level representation, which is needed to capture the dynamic patterns of input graphs. On the other hand, the node-level representation captures the local structure, while the graph-level representation encapsulates the global structure of the temporal graph. Therefore, for more accurate event prediction, both node-level and graph-level representations are indispensable.

Recently, self-supervised learning has demonstrated significant success across diverse domains, including natural

language processing [7], [8] and computer vision [9], [10]. A central concept in this framework, contrastive learning [11], aims to learn representations that preserve similarity by contrasting two or more semantically coherent views obtained through data augmentations of the same underlying object.

A more recent line of research [12]–[14] has explored extending the contrastive framework to graph representation learning and achieved superior performance to regular GNN models. While some models apply different graph augmentation methods, e.g., edge dropping, subsampling, etc., to create different views, some models create different graphs from different types of data, such as an attribute graph in addition to the original graph.

However, despite the high performance of contrastive learning on static graphs, there are only a few methods that use contrastive learning in dynamic graphs for downstream tasks like dynamic link prediction and node classification [15], [16]. These methods also mainly focused on the local structure of nodes to learn the node representation and ignored the global structure of the dynamic graphs. Existing models use different timestamp graphs as different views of the input graphs as positive and negative samples for contrastive learning. These views mainly represent the local structure of the temporal graphs and are used for learning node representation.

Addressing the aforementioned challenges, this paper introduces a **Dynamic Graph Contrastive Learning (DyGCL)** method that learns dynamic graph representation for future event prediction. In our model, we have two different dedicated encoders, a local and a global view encoder that learns the local structure and global structure of the input graph, respectively. Our local view encoder captures local neighborhood structural information of the nodes in each time snapshot graph using a Dynamic Graph Convolutional Network via passing node features from the previous time to the current time snapshot graph. In the last time step, it applies a pooling layer to generate one representation for the graph. Furthermore, the global view encoder captures the global structure of the input graphs using dynamic Graph Pooling. We update both graph representations from the two different encoders using contrastive learning, where we maximize the cosine similarity between them. Finally, our model combines both representations using an MLP layer and uses it for future event prediction. As a result, by combining both encoders, DyGCL learns a dynamic graph representation that captures local and global dynamic structural patterns of input graphs. To the best of our knowledge, our model is the first of its kind to combine both node-level and graph-level temporal dependency for learning graph representations using contrastive learning for event prediction.

The main contributions of our work can be outlined as follows:

- We present an innovative framework called Dynamic Graph Contrastive Learning (DyGCL), expressly developed for predicting events. The DyGCL framework is composed of a local view encoder and a global view encoder. The local view encoder in DyGCL is responsible

for learning the graph representation that captures the local structure of the graph in each time-stamp graph, considering the previous time-stamp representations. In contrast, the global view encoder infers a unified representation for each time-stamp graph and maintains all the previous time-stamp representations to learn the future one. We update both graph representations using contrastive learning, where we maximize the similarity between the two representations.

- DyGCL utilizes both semantic and structural features of the input graph, enhancing the overall effectiveness of the model. Contrary to prior works, which primarily focused on the local structure of input graphs, our approach considers both the local and global structures of input graphs to comprehend the temporal dependency for event forecasting via the local view encoder and the global view encoder, respectively.
- We compare the results of our model against several deep learning and GNN-based baseline models for event prediction. The experiments demonstrate that our model outperforms these baseline models.

The rest of this paper is organized as follows. In section II, we discuss the related work on event prediction and how GNN and Dynamic GNN are used for event prediction and Contrastive learning in GNN. In section III, we present the preliminary concepts for the problem formulation, Graph Neural Network for Static Graph, and Contrastive Learning. We explain our methodology for DyGCL and how to incorporate it for event prediction in section IV. In section V, we present our results on real-world datasets by comparing current baseline methods. Our final remarks with future work directions are found in section VI.

II. RELATED WORK

A. Event prediction

In the area of event prediction, there is a broad spectrum of real-world applications that have been explored, such as predicting political events [6], [17], forecasting election results [18], traffic analysis, trends in the stock market [19], and tracking disease outbreaks [20]. Initial approaches in this field often utilized traditional machine learning techniques. For instance, the use of linear regression [21] has been noted for its effectiveness in predicting the timing of future events by analyzing social media data frequency and volume. Furthermore, more complex methods involving paragraph embeddings [22] and the utilization of topic-specific keywords [4] have been investigated.

The advancements in event prediction have seen a notable shift from basic machine learning techniques to the adoption of sophisticated deep learning methods, particularly focusing on the temporal aspects of information diffusion. The application of deep neural networks, notably by Ma et al. [23], [24] using recurrent neural networks, and Liu et al. [25] combined convolutional and recurrent neural networks, exemplifies this advancement. These models excel in understanding the progression and nuances of events as they unfold, particularly

in digital and social media contexts. Xia et al. [26] further contributed to this field with a model focused on the detailed detection and segmentation of evolving event states, offering a more granular perspective on event dynamics. Despite these technological advancements, a common limitation persists in the predominant focus on semantic information of input data, potentially overlooking other vital aspects, such as contextual and non-semantic factors, which are crucial for a comprehensive and accurate prediction model.

B. Graph-based event prediction

Graph-based event prediction has seen significant advancements with the implementation of GNNs, which excel in structuring the relationships among words or entities into graphs. The DynamicGCN [5] exemplifies this approach by applying a static Graph Convolutional Network (GCN) to input graphs of each time epoch, initializing node features for each epoch with embeddings from the previous one, and starting with word embeddings at the initial time. This model updates node embeddings via a temporal layer, integrating the initial and GCN-derived embeddings, and employs a readout layer to convert these embeddings into a fixed-size vector for event prediction. Similarly, DyGED [6] focuses on learning graph-level representations for each epoch’s graph, updated using a recurrent neural network (RNN). This model applies a static GNN and global pooling for learning representations but does not utilize hierarchical graph pooling, which is crucial for capturing the graph’s global structure. DyGED then merges all snapshot graph embeddings into a single vector representation for event prediction. While both models mark important contributions to graph-based event prediction, leveraging GNNs to capture complex event dynamics, they also highlight areas for potential improvements, such as the integration of more advanced graph pooling techniques and enhanced processing of temporal information for more robust and accurate predictions.

C. Graph contrastive learning

Contrastive learning [11], a subset of self-supervised learning techniques, has seen widespread adoption in domains like image processing [9], [10] and natural language processing. This approach learns data representations by distinguishing between similar (positive) and dissimilar (negative) pairs of data points. The core principle is to align representations of similar samples closely in the latent space while ensuring that dissimilar samples are farther apart. This technique is particularly valuable in scenarios with limited or unlabeled data, as it leverages the inherent structure of the data to learn meaningful representations without relying on external labels.

Graph contrastive learning extends the principles of contrastive learning to the realm of graph representation learning. It addresses the unique challenges posed by graph-structured data, aiming to capture both local node-level and global graph-level structural information. Inspired by Deep InfoMax, Deep Graph Infomax (DGI) [27] maximizes the mutual information between a node’s representation and a high-level summary

of the entire graph for learning node representation. This maximization aids in overcoming the limitations of vanilla graph convolutional networks by incorporating a more comprehensive understanding of the graph’s global information. Methods like [13] generate two augmented views of a graph using diffusion matrix and original graph adjacency matrix and learn by bringing representations of the same node in different views closer together.

DDGCL [15] uses contrastive learning for dynamic graphs where it constructs temporal views by sampling nodes from k -hop neighborhood at different timestamps. On the other hand, CLDG [16] uses different timespan views of the same graph as a contrastive pair. Both methods mainly focus on the local structure of the graph to learn node representation. Despite challenges like computational intensity and sampling bias, graph contrastive learning has proven effective in unsupervised learning tasks on graphs, offering significant insights for tasks such as node and graph classification and recommendation systems.

Existing Dynamic GCN models or Graph Contrastive models either focus on node-level representation or graph-level representation for event prediction. As a result, they ignore the local or global dynamic structure of the input graphs. In our method, we address this problem and learn node-level and graph-level representation for each time epoch graph. While learning graph-level representations, we apply a hierarchical graph pooling method that helps to capture the global structure of the graph effectively.

III. PRELIMINARIES AND PROBLEM DESCRIPTION

In this section, we first present some primary concepts and terminology about dynamic graphs and contrastive learning and then define the event prediction problem.

For an event at a specific location, historical event-related articles reveal important information about the rising event. Therefore, these are used to predict future events. In this project, event-related articles are encoded into a sequence of graphs as a dynamic graph where each graph represents the contextual information from a specific timestamp. While nodes in the graph represent words occurring in the articles of that timestamp and edges between nodes represent the occurrence of the words in a predefined fixed-size window. One graph is created for each day and data from k consecutive days is represented as the dynamic graph and is used to learn and predict whether an event will occur on $(k + 1)$ th day.

Here, we first define dynamic graphs and then event prediction via dynamic graphs.

Definition 1 (Dynamic Graph): A dynamic graph \mathbb{G} is defined as a series of T discrete snapshots denoted as $\mathbb{G} = \{G_1, G_2, \dots, G_T\}$, where G_t represents the graph at timestamp t . Each G_t has an adjacency matrix A_t showing the relation between nodes at time t

We define the event prediction problem as the binary classification problem and predict whether there is an event on day $t + 1$ using data from t previous days. Formal definition of the event prediction via dynamic graphs is given as follows;

Definition 2 (Event Prediction): Given a training dataset \mathbb{D} , where each sample is represented as a dynamic graph $\mathbb{G} = \{G_1, G_2, \dots, G_T\}$ with initial node feature matrix $H_t \in \mathbb{R}^{N \times d}$ where N is the number of nodes with d dimension at time t , our goal is to learn a graph encoder that maps the input dynamic graph into vector representation and use this representation to predict the future event \hat{y} at time $T + 1$.

For our dynamic graph datasets, which are constructed from the text of event-related articles, we have one global initial node feature matrix H_{sem} representing the semantic meaning of all words appearing over the time obtained with a word embedding model. H_t can be obtained by filtering the words occurring at time t .

A. Graph Neural Network for Static Graph

Recently, GNNs have been used as benchmark models for static graphs for different types of downstream tasks including node classification, link prediction, and graph classification. Graph convolution network (GCN) [28] is the most widely used GNN model. GCN is a multilayer neural network to processes the graph data where it combines the features of each node from its neighbors while propagating the information through the edges. Given an input graph as $G(V, A, H)$ where V is the node-set, $A \in \mathbb{R}^{N \times N}$ is the adjacency matrix and $H \in \mathbb{R}^{N \times d}$ is the feature matrix with d dimensional node feature and N is the number of nodes in the graph GCN layer transform the node representation as follows:

$$H^{(1)} = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(0)} \theta^{(0)}) \quad (1)$$

where σ is a non linear activation function, $\tilde{A} = A + I$ is the adjacency matrix with self-loop, $\tilde{D} \in \mathbb{R}^{N \times N}$ is the normalized degree matrix of \tilde{A} , $\theta^{(0)}$ is trainable weight. GCN model uses multiple convolutional layers to learn the spatial feature for nodes from the connected nodes as follows:

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} \theta^{(l)}) \quad (2)$$

where $H^{(l)}$ feature matrix and $\theta^{(l)}$ model parameter for $l^{(th)}$ layer. The underlying intuition for each layer is that nodes gather and aggregate information from their local neighbors. With the l layers GCN model, nodes can get information from l -hops neighborhood information.

In general, while training the model for the classification problem, cross-entropy loss is used as the supervised loss function, which compares the actual label with the predicted label.

B. Contrastive Learning

Contrastive learning (CL) has become one of the most popular approaches for unsupervised representation learning, which learns through comparisons among different samples. This comparison is typically conducted between positive pairs of "similar" inputs and negative pairs of "dissimilar" inputs. Contrastive learning in the graph domain aims to learn node or graph representation for a given input graph by maximizing

the similarity between the different views of the input graph in their latent space via contrastive loss. In general, there are three parts to graph contrastive learning: i) View construction, ii) View encoder, and iii) Contrastive loss.

- **View Construction:** A view is represented as graph data, denoted as $vi = (A_i, X_i)$, where $A_i \in \mathbb{R}^{N \times N}$ and $H^{(i)} \in \mathbb{R}^{N \times d}$. While it is common to get 2 views, there are also some models that create more than 2 views [29], [30]. In practical terms, view augmentation approaches involve techniques like node dropping [31], edge perturbation [32], attribution masking [33], and subgraph sampling [34].
- **View Encoder:** View encoder maps the node representation or graph representation from different views of input graphs. As encoders for learning representations from views, any GNN models can be used based on the problem and type of graphs.
- **Contrastive loss:** After learning the representations for different views using the view encoder, the contrastive learning model is optimized by a contrastive loss. The objective of the contrastive loss function is to maximize the similarity among different view representations of the same graph and minimize similarity for view representations of other graphs. In general, the contrastive loss can be described as follows:

$$L_{contra} = -\log\left(\frac{\text{sim}(Z_{vi}, Z_{vj})}{\sum_{j'=1}^N \text{sim}(Z_{vi}, Z_{vj'})}\right) \quad (3)$$

where the $\text{sim}(\cdot)$ function measures the similarity between two representations. We can use any similarity function like cosine similarity and mutual information (MI) in the contrastive loss. Z_{vi} and Z_{vj} are the i th view and j th view representations of node v .

IV. METHOD

In this section, we present our proposed Dynamic Graph Contrastive Learning (DyGCL) model, including local view and global view encoder with a contrastive loss for event prediction. The overall architecture for the model is presented in Figure 1. As depicted in Figure 1 and Algorithm 3, our model is specifically designed to optimize dynamic graph representation learning, particularly for event prediction tasks. The model consists of three main components.

The first component is the local view encoder, where we use a Dynamic Graph Convolutional Network to extract local structural information from the dynamic graphs. Dynamic Graph Convolutional Network extracts dynamic node representations from temporal input graphs. For each temporal graph, it utilizes node embeddings from the preceding time step to enhance the embeddings of the current step. Following this, a temporal attention layer combines the original node embedding with the output of the current step to keep the semantic information in the learning process.

In the second component, we introduce a global view encoder where we use Dynamic Graph Pooling. Dynamic Graph Pooling generates a hierarchical representation of each

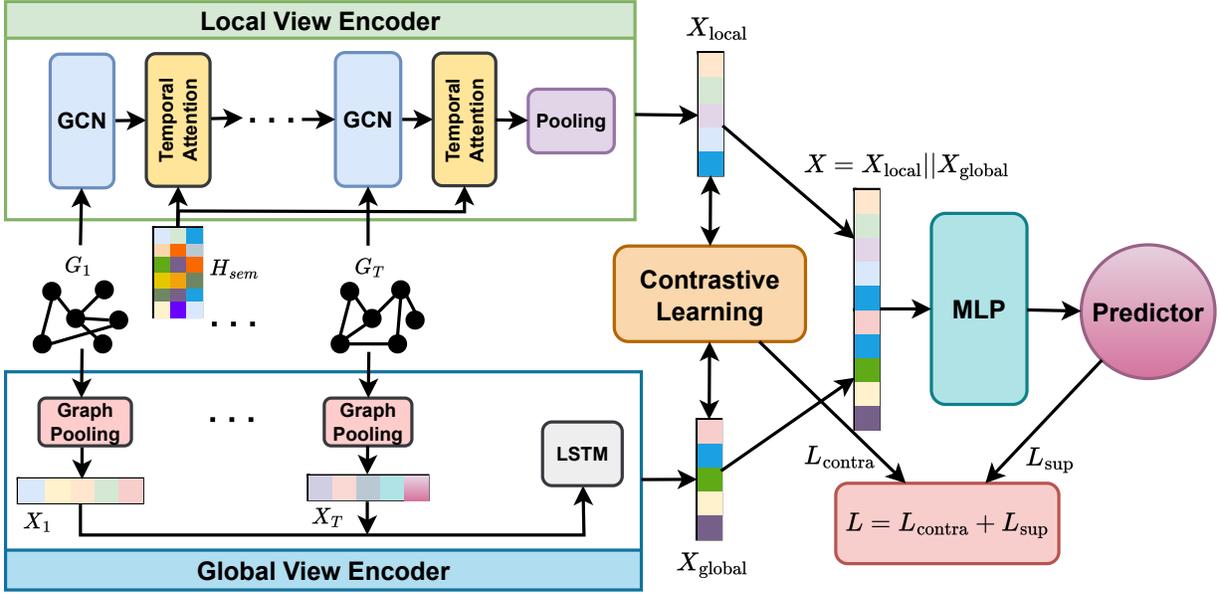


Fig. 1: An overview of Dynamic Graph Contrastive Learning, (DyGCL) architecture. We feed input graphs into the Local View Encoder to learn the dynamic node representations. In the end, node representations are pooled into a graph-level representation using a pooling layer. We also feed input graphs into the Global View Encoder to learn dynamic graph representation. Use Contrastive learning to maximize the similarity between two graph representations. Finally, representations from Local View Encoder and Global View Encoder are combined by an MLP layer and feed the representation to the predictor for event prediction.

temporal graph using a hierarchical pooling method. These temporal graph representations are then refined by a recurrent neural network, considering both the current and previous temporal graph representations.

The third component is contrastive learning, which helps update the representations from the two encoders to maximize the similarity between them. Then, these representations are combined and given to an MLP layer for event prediction.

A. Local View Encoder

In general, a graph-level representation is derived from the node-level representations of the graph. Therefore, obtaining an improved graph representation hinges on having superior node representations. To enhance node representation, it is crucial to consider the local structure of nodes, including neighborhood relationships and interactions with neighbors. To learn better node representation by preserving the local structure of the nodes and getting graph-level representation from the node-level representation, we introduce a local view encoder.

In the local view encoder, we use a Dynamic Graph Convolution Network (DyGCN) that learns dynamic node representation, preserving the temporal local structure for dynamic graphs. It passes the information from the previous time step to the next step to capture the dynamically changing neighboring structures among the input graphs. DyGCN comprises two layers: the static GCN layer and the temporal attention layer. For every time step graph, we apply the static GCN to learn node representations based on the current graph structure and

also node features coming from the previous time step. This allows us to encapsulate the local structure of nodes at each time step into vector representations. Within each time step, the static GCN learns the node representation using a message-

Algorithm 1: Dynamic Graph Contrastive Learning (DyGCL)

Input: Initial Node Features matrix $H_{(sem)}$, Temporal Graphs, Event Label Y

Output: Predicted Event label \hat{y}

$\mathcal{G} \leftarrow$ randomly sample a batch

for i in \mathcal{G} **do**

$A_{T-k, \dots, T}, H_{sem} = \mathbb{G}_i$

$Z_{local}, H_L = LVE(A_{T-k, \dots, T}, H_{sem})$

//Local view graph representation

$Z_{global} = GVE(A_{T-k, \dots, T}, H_L)$

//Global view graph representation

$L_{contra} = \frac{Z_{local} Z_{global}^T}{\|Z_{local}\| \|Z_{global}\|}$

//Contrastive Loss

$Z_{local} = Z_{local} W_l + b_l$

$Z_{global} = Z_{global} W_g + b_g$

$Z_{Final} = MLP([Z_{local} || Z_{global}])$

$\hat{Y} = \sigma(MLP(Z_{Final}))$

$\mathcal{L}_{sup} = -\sum Y \log \hat{Y}$

$L = \alpha \mathcal{L}_{sup} + (1 - \alpha) L_{contra}$

end

Algorithm 2: Local View Encoder (LVE)

Input: Initial Node Features matrix H_{sem} , Temporal Graphs.

Output: Local view Graph Representation Z_{local} , Updated node Features H

for each $t \leftarrow T - k$ to T **do**

$$H^{(t)} = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A}^t \tilde{D}^{-\frac{1}{2}} H^{(t-1)} \theta^{(t-1)})$$

//Temporal Attention

$$H^{(t)'} = H^{(t)} W_s^t + b_s^t$$

$$H_{(sem')} = H_{sem} W_0^t + b_0^t$$

$$H^{(t)} = \tanh([H^{(t)'} || H_{(sem')}])$$

end

$$Z_{local} = \frac{1}{N} \sum_{i=1}^N [h_i]$$

passing approach, which is described as

$$H^{(t)} = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(t-1)} \theta^{(t-1)}) \quad (4)$$

where $H^{(t)}$ represents the node representation matrix for the t^{th} snapshot graph, σ is an activation function, $\tilde{A} = A + I$ is the adjacency matrix with self-loop, $\tilde{D} \in \mathbb{R}^{N \times N}$ is the normalized degree matrix of \tilde{A} , $\theta^{(t)}$ is the trainable weight for t^{th} time step and $H^{(t-1)}$ is the input node representation matrix for t^{th} time steps obtained from previous time step's Dynamic GCN.

While the topological structure of the graph and its changes over time are important features to detect events, semantic meanings of the words are also important for events. On the other hand, when GCN is applied at each time epoch, it updates node representations with the current time epoch neighborhood. Over multiple timestamps, there is a risk of losing essential semantic details of nodes, potentially leading to the over-smoothing issue. To address this challenge, we create a temporal attention layer. We define initial node features, H_{sem} with their word embeddings, which represent their semantic meanings. After obtaining node representations from the current time epochs' GCN, we refine these representations through a temporal attention layer with initial semantic representations, H_{sem} . In this layer, we combine the node representation from the current timestamp's GCN and the initial semantic node features using a linear neural network layer. First, we multiply them by learnable weight matrices that signify the importance of each representation for event prediction. Then we concatenate weighted representations and give it to an activation function. Here we use \tanh as the activation function. By refining the node representation obtained from GCN with the initial semantic word embedding in each temporal attention layer, we facilitate the fusion of semantic attributes of the nodes. The temporal attention layer applied at each time step is formalized as follows:

$$\begin{aligned} H^{(t)'} &= H^{(t)} W_s^t + b_s^t \\ H_{(sem')} &= H_{sem} W_0^t + b_0^t \\ H^{(t)} &= \tanh([H^{(t)'} || H_{(sem')}]) \end{aligned} \quad (5)$$

Algorithm 3: Global View encoder (GVE)

Input: Node Features matrix H , Temporal Graphs.

Output: Global view graph representation Z_{global}

for each $t \leftarrow T - k$ to T **do**

$$S = \sigma(\text{GNN}(H^t, A^t, \theta_{att}^t))$$

$$idx = \text{topK}(S, [\alpha \times N])$$

$$A^{(t+1)} = A_{idx, idx}$$

$$Z_t = \frac{1}{N} \sum_{i=1}^N x_i || \max_{i=1}^N x_i$$

end

$$Z_{global} = \text{RNN}(Z_{T-k}, \dots, Z_T)$$

where $H^{(t)}$ is the embedding matrix from GCN layer at time t , $H^{(0)}$ is the initial node embeddings or pre-trained word embeddings, W_s^t , b_s^t and W_0^t , b_0^t are learnable parameters for $H^{(t)}$ and $H_{(sem')}$, respectively, and $||$ is the concatenation operation.

After updating node embeddings using the temporal attention layer, we pass it as initial node embedding to the next GCN layer at the time stamp $t+1$. After getting the final node representation from the Dynamic Graph Convolution layer at the last time step, we convert the node representations into graph representations using global mean pooling, which takes the average of node features defined as follows:

$$Z_{local} = \frac{1}{N} \sum_{i=1}^N [h_i] \quad (6)$$

where N is the number of nodes, h_i is the i^{th} node feature.

B. Global View Encoder

The local view encoder is primarily engineered to derive graph-level representation from the dynamic node-level representations from temporal graphs. However, this node-centric approach tends to capture only the local structures of input graphs, often sidelining the global structures. While local structures show close-by connections, global structures show bigger patterns across the entire graph. For predicting events, these big patterns are also important. Sometimes, events happen because of connections that aren't right next to each other. If we only focus on the local structure, we might miss these important clues. Thus, for predicting events, it is very important to consider global structures.

To capture higher-order features in input graphs, we introduce a global view encoder where we use a Dynamic Graph Pooling module. At first, we learn a hierarchical graph representation for each time snapshot graph. These graph representations are then refined by a recurrent neural network, considering both the current and previous temporal graph representations to capture relations between them and also to capture the changes from one to another.

To learn the hierarchical graph representation, selection-based and clustering-based pooling models can be applied. Here, we employ top- K pooling as a selection-based graph pooling method [35]. Selection-based methods are chosen for their memory efficiency and emphasis on the global structure

of the input graph. In top- K pooling, we select the top k important nodes from the input graph that are deemed relevant to the event and keep these nodes and relations between them for the next layer. The selection of the top k nodes is determined by calculating attention scores for all nodes, leveraging node features, the GNN model, and an attention parameter, as follows:

$$S = \sigma(\text{GNN}(H^t, A^t, \theta_{att}^t)) \quad (7)$$

Where $S \in \mathbb{R}^{N \times 1}$ represents the node attention scores, H^t denotes the node embeddings at time t , A^t is the adjacency matrix, $\theta_{att}^t \in \mathbb{R}^{d \times 1}$ is the learnable parameter matrix at time t , and N is the number of nodes.

Following the calculation of attention scores, we select the top k nodes with the highest scores. Subsequently, we construct a new coarse graph using the selected nodes, as outlined below:

$$\begin{aligned} idx &= \text{topK}(S, [\alpha \times N]) \\ A^{(l+1)} &= A_{idx, idx} \end{aligned} \quad (8)$$

where idx is the indices of top- k nodes, α is the pooling ratio, N is the number of nodes, and $A^{(l+1)}$ is the coarse graph. Multiple graph pooling and GCN layers on each snapshot graph are applied to get a hierarchical graph representation.

As the final layer of the hierarchical graph representation learning, we apply a readout layer to get a fixed-sized graph-level representation for each snapshot graph. The readout function aggregates the node features as follows:

$$Z_t = \frac{1}{N} \sum_{i=1}^N [h_i || \max_{i=1}^N h_i] \quad (9)$$

where N is the number of nodes, h_i is the i^{th} node feature and $||$ denotes concatenation. The graph pooling method effectively captures the static global structure of the current time epoch graph. Yet, to grasp the dynamic evolution of the global structure in temporal graphs, it is crucial to update the current time epoch graph representation with that of the previous time epoch graphs. For this purpose, a Recurrent Neural Network (RNN) layer is employed. RNNs, well-suited for sequential data, update the current data representation by considering the input data from the preceding steps in the sequence. In our approach, we feed graph representations from the pooling layers into the RNN layer in the following manner:

$$Z_{global} = \text{RNN}(Z_{t-k}, \dots, Z_t) \quad (10)$$

where z_{t-k} is the graph-level representation of k -previous time steps from current time graph. The output of the RNN layer is used as the global graph features.

C. Contrastive Learning

While the local view encoder and global view encoder extract the different features of the dynamic graph, their final representation should be similar as they belong to the same data. Therefore, we apply contrastive learning to make them similar. Our contrastive learning objective function aims

to maximize the cosine similarity or minimize the cosine distance between the graph representations from local and global view encoders. In contrastive learning, while one view should be central, we should define positive and negative samples to compare and measure the similarity. We use graph representations from the local view encoder and the global view encoder as positive samples if they belong to the same data. In our model, we do not use any negative sample in contrastive learning as Namkyeong Lee et, al [12] mentioned that contrastive learning on graphs performs better without negative samples. As our objective, we minimize the cosine distance between positive samples. We define our objective function as follows:

$$L_{\text{contra}} = \frac{Z_{\text{local}} Z_{\text{global}}^T}{\|Z_{\text{local}}\| \|Z_{\text{global}}\|} \quad (11)$$

Where Z_{local} is the graph-level representation from local view encoder and Z_{global} is the graph-level representation from global view encoder

D. Output layer

In the local view encoder, we derive a dynamic node-level representation through the Dynamic GCN module, that predominantly encapsulates the local intricacies of temporal graphs. Following this extraction, we introduce a pooling layer at the culmination of the Dynamic GCN. This layer is responsible for morphing the node embedding into a standardized graph-level embedding, as depicted in Equation 9. Concurrently, from the global view encoder, we utilize a pooling technique to obtain a uniform graph representation, encoding the overarching structure of the graph. In our approach, the graph-level outputs from both the local view encoder and global are integrated, allowing us to embed both temporal local and global graph structural information into a singular graph-level embedding. The amalgamation of these embeddings is facilitated by an MLP layer. To combine local and global view representations, we first employ two distinct learnable weights for the two embeddings, allocating specialized attention to each. Once the embeddings are individually multiplied by their respective learnable weights, they are concatenated. Subsequently, an activation function is employed to seamlessly merge them, as outlined below

$$\begin{aligned} Z_{\text{local}} &= Z_{\text{local}} W_l + b_l \\ Z_{\text{global}} &= Z_{\text{global}} W_g + b_g \\ Z_{\text{Final}} &= \text{tanh}([Z_{\text{local}} || Z_{\text{global}}]) \end{aligned} \quad (12)$$

where Z_{local} is graph embedding matrix from local view encoder, Z_{global} is graph embedding from global view encoder, W_l, b_l , and W_g, b_g are learnable parameters for Z_{local} and Z_{global} , respectively, $||$ is the concatenation operation and tanh is an activation function of the linear layer.

After combining both graph representations into one final graph representation, our model gives it as an input to a multi-layer perception layer with the sigmoid function to predict the event occurrence and calculate the supervised loss as follows:

TABLE I: Performance comparisons of our model with baseline models on event prediction

Method	Model	Thailand	Egypt	Russia	India	NYC Cab	Twitter Weather
Static	GCN	76.13	75.8	76.63	67.45	84.91	77.21
	TopKpool	77.03	85.28	78.45	65.53	86.45	78.65
	SAGPool	77.74	86.12	80.86	68.50	90.82	80.78
	DiffPool	76.13	82.8	79.63	67.48	88.70	76.6
Dynamic	GCN+GRU	79.28	83.88	79.66	67.48	85.00	76.50
	GCN+LSTM	78.13	83.05	79.38	68.10	85.07	76.55
	EvolveGCN	-	-	-	-	84.20	78.24
	DynamicGCN	80.92	84.71	84.71	68.70	81.00	71.30
	DyGED	73.50	85.41	81.43	68.88	91.20	81.00
	DyGCL (Ours)	86.57	89.28	88.95	76.85	95.80	90.68

TABLE II: Dataset statistics. $|S|$ is the number of samples, \bar{N} and \bar{E} are the average number of nodes and edges, respectively, and $|Ev|$ is the number of events.

Datasets	$ S $	\bar{N}	\bar{E}	$ Ev $
Thailand	1883	600	7281	715
Egypt	3788	675	9680	1469
Russia	3552	645	9776	1171
India	12249	685	12994	4586
NYC Cab	4464	263	3717	162
Twitter	2557	1000	10312	287

$$\hat{Y} = \sigma(MLP(Z_{Final}))$$

$$\mathcal{L}_{sup} = - \sum Y \log \hat{Y} \quad (13)$$

where Z_{Final} is the graph representation, \hat{Y} is the predicted event occurrence, and Y is the actual event occurrence.

We jointly train our model with a weighted sum of the supervised loss and contrastive loss as follows:

$$L = \alpha L_{sup} + (1 - \alpha) L_{contra} \quad (14)$$

where α is a hyperparameter of the model.

V. EXPERIMENT

In assessing our model’s performance for the event prediction task, treating it as a dynamic graph classification problem, we conduct a comprehensive evaluation. We compare our model’s performance against six distinct baseline models. Additionally, we delve into the analysis of the impact of the number of historical days on event prediction. Furthermore, we present results for variations of our model, incorporating different message-passing models and graph-pooling methods. We also visualize the global structure of temporal graphs.

A. Datasets

In our experiments, we use six datasets. Among them, four of them are social event datasets, one is a weather event dataset and the other is a traffic event dataset. Table II shows the statistics of six datasets.

Thailand, Egypt, Russia, and India event datasets are collected from the Integrated Conflict Early Warning System

(ICEWS) [5], [36]. These datasets include information on political events and are designed for assessing both national and international crises. We concentrate on data sourced from major cities such as Delhi, Mumbai, Kolkata, and others in India, Bangkok in Thailand, Cairo in Egypt, and Moscow in Russia. Rallies, strikes, violent protests, and passage obstructions are frequent event types on these datasets.

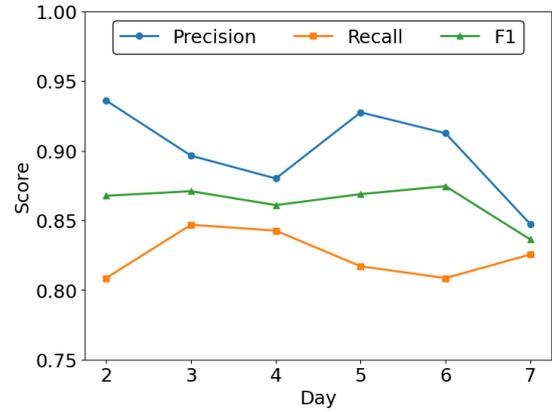
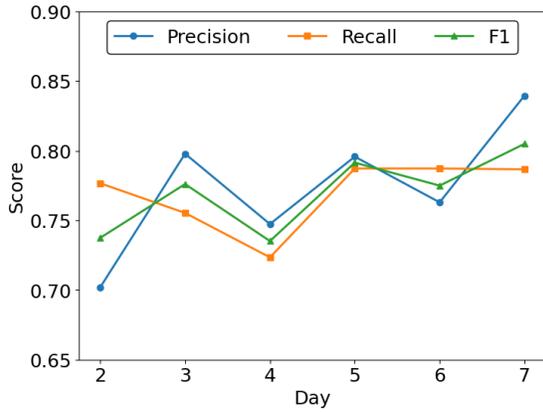
The **NYC Cab** represents a mobility network that contains geo-tagged mass-gathering events, such as concerts and protests. The **Twitter Weather** dataset showcases user-generated content pertaining to weather events, including storms and earthquakes.

B. Baseline

We use several graph neural network methods as baseline methods which mainly focus on Dynamic graphs or static graph classification. We divide our baseline methods into two classes: (1) Static methods, and (2) Dynamic methods.

1) *Static Methods*: We compare our model with GCN [28], TopKpool [37], SAGPool [35] and DiffPool [38], which are most common graph representation learning methods. For the static methods, we combine all time step graphs in a dynamic graph into one graph for each sample.

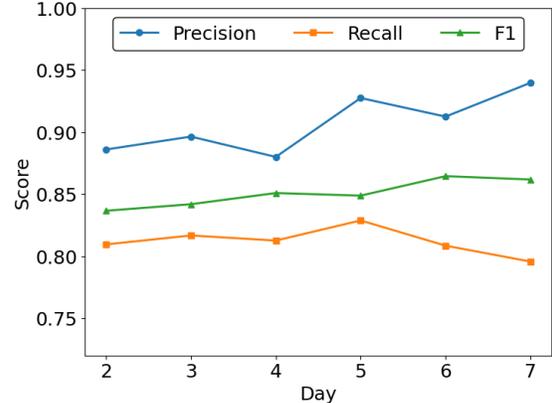
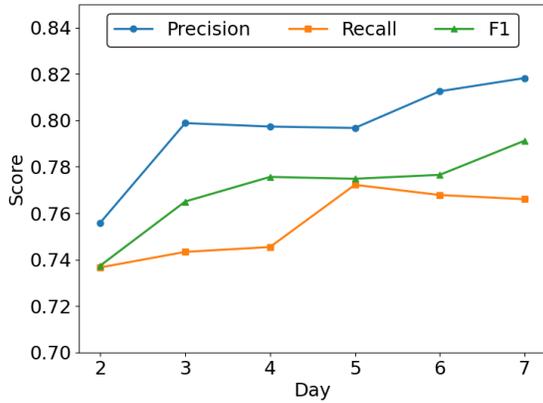
- *GCN*: Graph Convolutional network is the most popular GNN model for node representation learning. GCN aggregates the neighborhood information to update the node representation. After getting node representation we apply global pooling described in equation 6 to get graph-level representation.
- *TopKPool*: TopKPool is a hierarchical graph pooling method that selects top- k nodes for pooling operation. It considers the topological structure of the graph to select the top- k nodes.
- *SAGPool*: SAGPool is also a hierarchical graph pooling method that selects top- k nodes using the self-attention method to select top- k nodes for pooling operation.
- *DiffPool*: DiffPool is a cluster-based hierarchical graph pooling method. It uses a Graph Neural Network to learn a cluster assignment matrix of the input graph. Then it uses the cluster assignment matrix for graph pooling where it combines the nodes in each cluster into supernodes for the next layer.



(a) Thailand

(b) Egypt

Fig. 2: Detailed Event prediction results of $DyGCL_{sup}$ model without contrastive learning for the different number of historic days.



(a) Thailand

(b) Egypt

Fig. 3: Detailed Event prediction results of $DyGCL$ model for the different number of historic days.

2) *Dynamic Methods*: For the dynamic baseline models, we use GCN+LSTM [39], GCN+GRU [39], DynamicGCN [5], EvolveGCN [40] and DyGED [6].

- *GCN+LSTM*: This model learns the dynamic graph representation using GCN and LSTM models. GCN+GRU is a variation of this model where it replaces LSTM with the GRU model. It is designed as a temporal graph neural network to predict traffic conditions in the traffic network.
- *DynamicGCN*: This model applies GCN and a temporal layer for each snapshot to learn dynamic node representations. It converts dynamic node representation to graph representation using a mask linear layer. It is designed for event prediction on social media data.
- *EvolveGCN*: EvolveGCN is a dynamic graph convolution network that uses GCN and Recurrent Neural Network (RNN) to learn the representation of dynamic networks. It transfers the parameter matrix from one timestamp's GCN to another timestamp's GCN. It is a more general model that is applied to edge prediction, edge classification, and node classification for dynamic graphs. Here

we also use the global pooling method in the last layer of the model to convert node representation to graph representation.

- *DyGED*: DyGED is the most recent event prediction model that uses global graph pooling on each time step graph and applies RNN models to include macro-level graph dynamics for graph representation learning.

C. Experimental Settings

To assess our model for the event prediction task, we partition the data into three segments: 70% for training, 15% for validation, and 15% for testing. This splitting process is repeated ten times using ten different random seeds. We take the average of 10 different runs as the final results. Our model is implemented using PyTorch and PyTorch Geometry library, with the optimization performed by the Adam Optimizer. In order to determine optimal hyperparameters, we conduct grid search within the specified ranges: learning rate in $\{1e-2, 5e-2, 1e-3, 5e-3, 1e-4, 5e-4\}$, weight decay

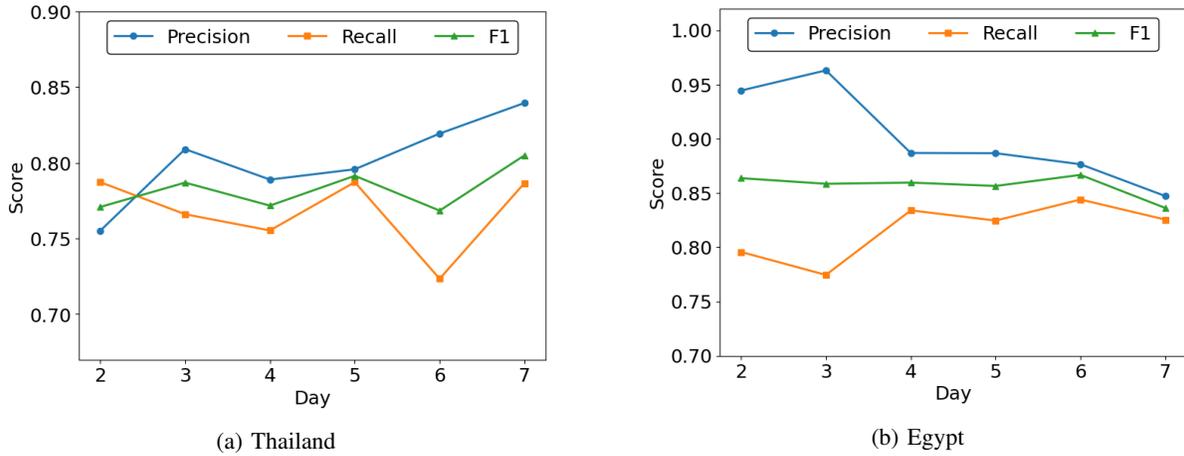


Fig. 4: Detailed event prediction results of $DyGCL_{sup}$ model without contrastive loss for the different number of lead days.

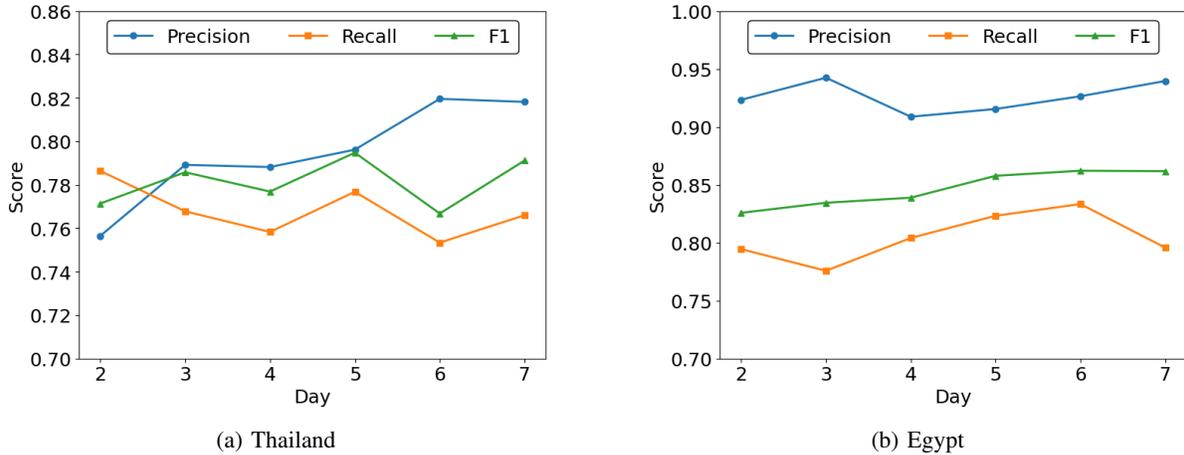


Fig. 5: Detailed event prediction results of the $DyGCL$ model for the different number of lead days.

in $\{1e-2, 1e-3, 1e-4, 1e-5\}$, pooling ratio in $\{1/2, 1/4\}$, dropout ratio fixed at 0.2, and hidden size in $\{16, 32, 64, 128\}$.

The training process halts if the validation loss fails to improve for 50 epochs. For the initial node embeddings of the dynamic graph, we use a 100-dimensional word embedding vector obtained through the Word2Vec method.

D. Result

In Table I, we present a summary of our event prediction accuracy results, with results of Static and Dynamic baseline models. All models were run ten times with ten random seeds, and average accuracy was calculated for event prediction. We treat the event prediction task as a binary graph classification problem, where class labels are $\{0, 1\}$ indicating whether an event occurs on that day or not. The EvolveGCN model is applicable to datasets where all samples have the same number of nodes. This constraint limits the availability of results to only NYC Cab and Twitter weather datasets that meet this requirement. For other datasets, the number of nodes is different for different samples.

As we can see from Table I, our model consistently works well and outperforms all baseline models across all datasets. Notably, our model improves the highest accuracy of baselines by 6.9% for the Thailand dataset with 86.57% accuracy, 3.1% for Egypt with 89.95% accuracy, 4.2% for Russia with 88.96% accuracy, 11.5% for India datasets with 76.85 accuracy, 4.6% for NYC Cab dataset with 95.80 accuracy and 9.6% for Twitter Weather dataset with 90.68 accuracy. For political datasets (Thailand, Egypt, Russia, India), the DynamicGCN model achieves the second-highest accuracy, except for the Egypt dataset where SAGPool yields the second-highest accuracy. For NYC Cab and Twitter weather datasets, DyGED gets the second-highest accuracy.

Additionally, we observe that dynamic methods generally outperform static baseline methods for all datasets, with the exception of the Egypt dataset where SAGPool, as the static graph pooling method, performs better than other baselines. This result suggests the importance of global structural information for the Egypt dataset. Importantly, the graph pooling method consistently demonstrates superior accuracy for these

two datasets, indicating the significance of higher-order structural information.

Sensitivity Analysis: In our previous experiment, we used the previous 7 days’ data to make a prediction on Day 8. In this experiment, we investigate the prediction performance by varying both the number of prior days and lead time. To see the effect of contrastive learning on prediction performance for different numbers of days time we run our original model DyGCL and DyGCL_{sup} where we directly combine the representations of local and global view encoders using the MLP layer. We remove the contrastive loss L_{contra} from the model and use only the supervised loss L_{sup} to optimize the model.

Historical days denote the previous number of days used as the training data. We want to see whether there is an effect of the number of historical days preceding the event on the event prediction task by changing it from 2 to 7. In Figure 2 and Figure 3, we present precision, recall, and F1-score for different numbers of days for the event using DyGCL and DyGCL_{sup} models. In these figures, “day2” denotes the use of data from the last two preceding days (6th and 7th days) for predicting events on the 8th day. Similarly, for “day3”, we use 5th, 6th, and 7th days, and for other days. Contrary to the assumption that training with more days always leads to better results, figure 2 shows that this is not always true. Figure 2 shows the result for the DyGCL_{sup} model on the Thailand and Egypt datasets. From the figure, we can see that for the Thailand dataset, we get better results when we include all 7 days of information. We can also observe that using fewer days also gives more accuracy than a higher number of days. For example, for “day 3” where we use 3 days of information for event prediction, we get more accuracy than “day 4” where we use 4 days of information for event prediction. In contrast, better results are obtained with a fewer number of historic days in the case of the Egypt dataset. We get the highest score for “day 2” while we get the lowest score for “day 7”.

Figure 2 shows the result for the DyGCL model for the Thailand and Egypt datasets. For this model, we can see that for both datasets the Precision and F1-score increase gradually increase when we increase the number of historic days for event prediction. For both datasets, we get the highest score for “day 7”.

TABLE III: Performance comparisons of different GNN models on DyGCL for Thailand and Egypt datasets.

Dataset	Method	Precision	Recall	F1
Thailand	GCN	0.8182	0.766	0.7912
	GAT	0.75	0.798	0.773
	GraphSAGE	0.779	0.787	0.783
Egypt	GCN	0.9397	0.7957	0.8618
	GAT	0.834	0.855	0.814
	GraphSAGE	0.864	0.835	0.847

The lead time indicates the number of days in advance a model makes predictions of an event. In our experiment, we also explore the effect of the lead day on the model performance by varying the number of lead days from 1 to

7. We present precision, recall, and F1-score for different numbers of days for the event using DyGCL and DyGCL_{sup} models in Figure 4 and Figure 5. The lead day signifies how many days in advance our model predicts the event. As an example, when we refer to “Day6,” it implies the utilization of data from the 1st day to the 6th day for predicting the event on the 8th day, indicating a two-day advance prediction. In both datasets, we observe that the optimal lead time varies for different scoring metrics. From figure 4, we can see that the DyGCL_{sup} model provides the highest scores for Precision, Recall, and F1-score with “day 7” for the Thailand dataset. For the Egypt dataset, we get the highest Precision score for “day 2” and for Recall and F1-score we get the highest score for “day 6”. We can see that for all three scores, our model gives the lowest score for “day 7”. Notably, for all three scores, our model shows the lowest score for “day 7,” suggesting that adding more days’ information may introduce irrelevant details to the event.

In Figure 5, the performance of the DyGCL model with contrastive loss is illustrated for different numbers of lead days. Notably, for Precision and F1-score, DyGCL with “day 7” yields the highest scores for both datasets. In this scenario, DyGCL performs better with a higher number of lead days’ information compared to fewer days. For the Thailand dataset, we get the highest Recall score for “day 5” and for The Egypt dataset DyGCL gives the highest Recall score for “day 6”.

TABLE IV: Performance comparisons of different graph pooling models on DyGCL for Thailand and Egypt datasets.

Dataset	Method	Precision	Recall	F1
Egypt	Top- K	0.839	0.787	0.805
	SAGPool	0.8182	0.766	0.7912
	DiffPool	0.8161	0.7553	0.7845
Egypt	Top- K	0.847	0.826	0.836
	SAGPool	0.9397	0.7957	0.8618
	DiffPool	0.8664	0.8553	0.8608

Ablation Study: In our ablation study, we investigate the impact of different parts of the model including contrastive learning, the GNN model in the local view, the RNN layer, and the pooling layer in the global view, on the model performance. We present results for all ablation studies for Thailand and Egypt datasets as the representative except contrastive learning where we present results for all datasets.

We first experiment with GNN models. In addition to GCN as the default model, We apply two other popular GNN models, which are GAT and GraphSage to see the effect of them on the model performance. Table III shows the Precision, Recall, and F1 scores for different GNN models we use in our model to learn node representation for each snapshot graph in the local view. From the table, we can see that the GNN model to learn the node representation has a big impact on the model performance. For both datasets, the GCN model performs much better than the other models, especially with respect to precision. Also, the GraphSAGE model gives better results than the GAT model.

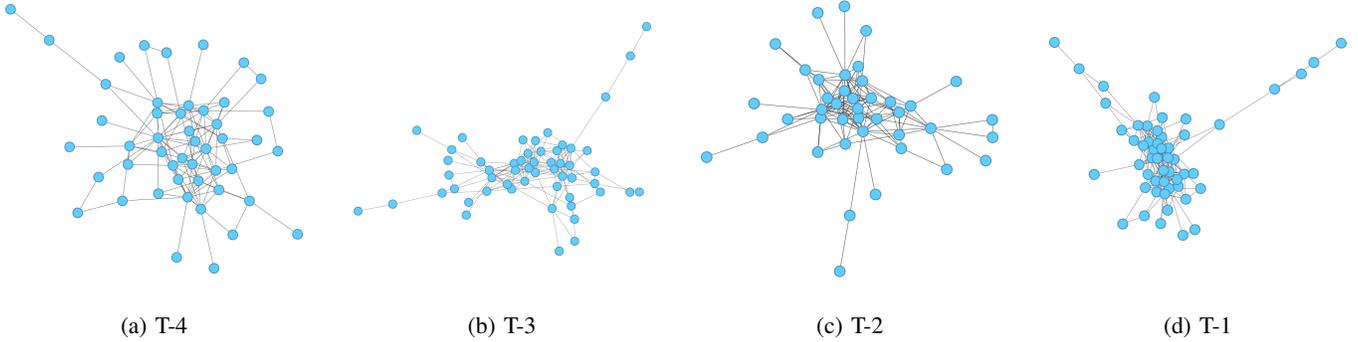


Fig. 6: Temporal pooled graphs in Global View Encoder for a sample event in NYC cab dataset.

TABLE V: Performance comparisons of different RNN models on DyGCL for Thailand and Egypt datasets.

Datasets	Method	Precision	Recall	F1
Thailand	LSTM	0.8182	0.766	0.7912
	GRU	0.8514	0.6702	0.75
	Transformer	0.7765	0.7021	0.7374
Egypt	LSTM	0.9397	0.7957	0.8618
	GRU	0.9151	0.8255	0.8680
	Transformer	0.9112	0.8298	0.8686

We also change the hierarchical graph pooling method in the global view encoder to see the effect of different graph pooling methods. In addition to Top- K pooling as the default one, we select SAGPool as the selection-based graph pooling method and DiffPool as the cluster-based pooling method. Table IV shows the Precision, Recall, and F1-score for three different graph pooling methods for the Thailand and Egypt datasets. From the table, we can see that the pooling method has a big impact on the model performance. In general, selection-based methods outperform the cluster-based method. For the Thailand dataset, the Top- k graph pooling method gives the highest accuracy and for the Egypt dataset, the SAGPool performs better than other methods.

In addition, we use different Recurrent Neural Networks(RNN) in the global view encoder to see their effect on them for the event prediction task. While LSTM is the original RNN model in the global view, we use two different popular RNN models, which are GRU, and Transformer in the global view encoder. Table V shows the Precision, Recall, and F1 scores for different RNN models. As we can see from the table, RNN model has an impact on the model performance, especially for the Thailand dataset. For the Thailand dataset, LSTM gives the highest Recall and F1 scores with around 0.6 increase. GRU gives the highest Precision value. On the other hand, the Transformer model performs better than the LSTM and GRU models for the Egypt dataset where it gives the highest Recall and F1 score. However, the difference is not high enough to say that it is the best model.

Furthermore, we investigate the impact of contrastive loss

TABLE VI: Performance comparisons with contrastive loss (DyGCL) and without contrastive loss (DyGCL_{sup}).

Dataset	DyGCL	DyGCL _{sup}
Thailand	86.57	85.87
Egypt	89.28	86.64
Russia	88.95	88.37
India	76.71	76.85
NYC Cab	90.68	89.88
Twitter weather	95.80	95.74

on event prediction. We run DyGCL and DyGCL_{sup} for all datasets and report the result in Table VI. Different than other ablation studies, we present results for all datasets. As we can see from the table, the model with contrastive loss DyGCL gives better accuracy than the model without contrastive loss DyGCL_{sup}. For Thailand, Egypt, NYC Cab, and Twitter weather dataset DyGCL model gives around 1%, 3%, 1%, 1%, and 3% more accuracy respectively than the DyGCL_{sup} model. For the Russia and India datasets, the accuracy of both models is very close where the difference is less than 1%.

Visualization: We also visualize the temporal graphs to see the global structure of the temporal graph before an event. For this experiment, we select a sample from the NYC Cab dataset where an event has occurred. We took four days' temporal graph before that event and applied our pre-trained DyGCL model. Then we get the pooled graph from the output of hierarchical graph pooling layers in the global view encoder and present the graphs in Figure 6. As we can see from the figure, the global structure of temporal graphs before the event is changing a lot and our model is capturing those structures with the local and global view encoders.

VI. CONCLUSION AND FUTURE WORK

In this work, we propose a Dynamic Graph Contrastive Learning model DyGCL for event prediction. There are two view encoders in our model one is a local view encoder and another one is a global view encoder. The local view encoder learns dynamic graph representation by capturing the

temporal local structure of input graphs and the global view encoder learns dynamic graph representation that encodes the temporal global structure of the input graph. We update both representations using contrastive learning where the objective is to maximize the similarity between two representations. Then our model combines the representations from the local view encoder and the global view encoder into a fixed-sized vector representation to capture dynamic local and global patterns among the temporal graphs. Finally, our model predicts events using this fixed-sized vector representation. In the experiment, we show that our model outperforms existing methods on event prediction tasks. In the future, we will increase the number of datasets and add more baseline models for comparison. We will also apply our model to different types of event prediction tasks like rumor detection, traffic event prediction, and health event prediction.

REFERENCES

- [1] N. Ramakrishnan, P. Butler, S. Muthiah, N. Self, R. Khandpur, P. Saraf, W. Wang, J. Cadena, A. Vullikanti, G. Korkmaz *et al.*, “beating the news’ with embers: forecasting civil unrest using open source indicators,” in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 1799–1808.
- [2] S. Deng, S. Wang, H. Rangwala, L. Wang, and Y. Ning, “Cola-gnn: Cross-location attention based graph neural networks for long-term ili prediction,” in *Proceedings of the 29th ACM international conference on information & knowledge management*, 2020, pp. 245–254.
- [3] A. Signorini, A. M. Segre, and P. M. Polgreen, “The use of twitter to track levels of disease activity and public concern in the us during the influenza a h1n1 pandemic,” *PLoS one*, vol. 6, no. 5, p. e19467, 2011.
- [4] X. Wang, M. S. Gerber, and D. E. Brown, “Automatic crime prediction using events extracted from twitter posts,” *SBP*, vol. 12, pp. 231–238, 2012.
- [5] S. Deng, H. Rangwala, and Y. Ning, “Learning dynamic context graphs for predicting social events,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 1007–1016.
- [6] M. Kosan, A. Silva, S. Medya, B. Uzzi, and A. Singh, “Event detection on dynamic graphs,” *arXiv preprint arXiv:2110.12148*, 2021.
- [7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.
- [8] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, “Language models are few-shot learners,” *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.
- [9] S. Gidaris, P. Singh, and N. Komodakis, “Unsupervised representation learning by predicting image rotations,” *arXiv preprint arXiv:1803.07728*, 2018.
- [10] M. Noroozi and P. Favaro, “Unsupervised learning of visual representations by solving jigsaw puzzles,” in *European conference on computer vision*. Springer, 2016, pp. 69–84.
- [11] T. Chen, S. Kornblith, M. Noroozi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.
- [12] N. Lee, J. Lee, and C. Park, “Augmentation-free self-supervised learning on graphs,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 7, 2022, pp. 7372–7380.
- [13] K. Hassani and A. H. Khasahmadi, “Contrastive multi-view representation learning on graphs,” in *International conference on machine learning*. PMLR, 2020, pp. 4116–4126.
- [14] F.-Y. Sun, J. Hoffman, V. Verma, and J. Tang, “Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization,” in *International Conference on Learning Representations*, 2020. [Online]. Available: <https://openreview.net/forum?id=r1Hf2NYvH>
- [15] S. Tian, R. Wu, L. Shi, L. Zhu, and T. Xiong, “Self-supervised representation learning on dynamic graphs,” in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 1814–1823.
- [16] Y. Xu, B. Shi, T. Ma, B. Dong, H. Zhou, and Q. Zheng, “Cldg: Contrastive learning on dynamic graphs,” in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, 2023, pp. 696–707.
- [17] H. Mueller and C. Rauh, “Reading between the lines: Prediction of political violence using newspaper text,” *American Political Science Review*, vol. 112, no. 2, pp. 358–375, 2018.
- [18] A. Tumasjan, T. Sprenger, P. Sandner, and I. Welp, “Predicting elections with twitter: What 140 characters reveal about political sentiment,” in *Proceedings of the international AAAI conference on web and social media*, vol. 4, no. 1, 2010, pp. 178–185.
- [19] J. Bollen, H. Mao, and X. Zeng, “Twitter mood predicts the stock market,” *Journal of computational science*, vol. 2, no. 1, pp. 1–8, 2011.
- [20] C. D. Corley, L. L. Pullum, D. M. Hartley, C. Benedum, C. Noonan, P. M. Rabinowitz, and M. J. Lancaster, “Disease prediction models and operational readiness,” *PLoS one*, vol. 9, no. 3, p. e91989, 2014.
- [21] Z. Xiao-Jun, “Twitter mood predicts the stock market,” *Journal of Computational Science*. 1–8., 2011.
- [22] Y. Ning, S. Muthiah, H. Rangwala, and N. Ramakrishnan, “Modeling precursors for event forecasting via nested multi-instance learning,” in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1095–1104.
- [23] J. Ma, W. Gao, Z. Wei, Y. Lu, and K.-F. Wong, “Detect rumors using time series of social context information on microblogging websites,” in *Proceedings of the 24th ACM international conference on information and knowledge management*, 2015, pp. 1751–1754.
- [24] J. Ma, W. Gao, P. Mitra, S. Kwon, B. J. Jansen, K.-F. Wong, and M. Cha, “Detecting rumors from microblogs with recurrent neural networks,” 2016.
- [25] Y. Liu and Y.-F. Wu, “Early detection of fake news on social media through propagation path classification with recurrent and convolutional networks,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.
- [26] R. Xia, K. Xuan, and J. Yu, “A state-independent and time-evolving network for early rumor detection in social media,” in *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, 2020, pp. 9042–9051.
- [27] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, “Deep graph infomax,” in *International Conference on Learning Representations*, 2019. [Online]. Available: <https://openreview.net/forum?id=rklz9iAcKQ>
- [28] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” ser. In *Proceedings of the ICLR*, 2017.
- [29] J. Chen and G. Kou, “Attribute and structure preserving graph contrastive learning,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 6, 2023, pp. 7024–7032.
- [30] U. Fang, J. Li, N. Akhtar, M. Li, and Y. Jia, “Gomic: Multi-view image clustering via self-supervised contrastive heterogeneous graph co-learning,” *World Wide Web*, vol. 26, no. 4, pp. 1667–1683, 2023.
- [31] J. Zeng and P. Xie, “Contrastive self-supervised learning for graph classification,” in *Proceedings of the AAAI conference on Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 10824–10832.
- [32] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, “Deep graph contrastive representation learning,” *arXiv preprint arXiv:2006.04131*, 2020.
- [33] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, “Graph contrastive learning with augmentations,” *Advances in neural information processing systems*, vol. 33, pp. 5812–5823, 2020.
- [34] S. Zhang, Z. Hu, A. Subramonian, and Y. Sun, “Motif-driven contrastive learning of graph representations,” *arXiv preprint arXiv:2012.12533*, 2020.
- [35] J. Lee, I. Lee, and J. Kang, “Self-attention graph pooling,” in *International Conference on Machine Learning*, vol. 97, 2019.
- [36] E. Boschee, J. Lautenschlager, S. O’Brien, S. Shellman, J. Starz, and M. Ward, “Icews coded event data,” 2015. [Online]. Available: <https://dataverse.harvard.edu/citation?persistentId=doi:10.7910/DVN/28075>
- [37] H. Gao and S. Ji, “Graph u-nets,” in *international conference on machine learning*. PMLR, 2019, pp. 2083–2092.
- [38] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, “Hierarchical graph representation learning with differentiable pooling,” *NeurIPS*, vol. 31, 2018.

- [39] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, and H. Li, "T-gcn: A temporal graph convolutional network for traffic prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 9, pp. 3848–3858, 2020.
- [40] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. Schardl, and C. Leiserson, "Evolvegcn: Evolving graph convolutional networks for dynamic graphs," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 04, 2020, pp. 5363–5370.