Multi-scale graph capsule with influence attention for information cascades prediction

Chen, X.; Zhang, F.L.; Zhou, F.; Bonsangue, M.M.

Received: 3 September 2021 | Accepted: 10 December 2021

DOI: 10.1002/int.22786

RESEARCH ARTICLE

WILEY

Multi-scale graph capsule with influence attention for information cascades prediction

Xueqin Chen[1,2] | **Fengli Zhang**[1] | **Fan Zhou**[1] | **Marcello Bonsangue**[2]

[1]School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu, China

[2]Leiden Institute of Advanced Computer Science, Leiden University, Leiden, The Netherlands

Correspondence
Fan Zhou, School of Information and Software Engineering, University of Electronic Science and Technology of China, No.4, Section 2, North Jianshe Rd, 610054 Chengdu, China.
Email: fan.zhou@uestc.edu.cn

Funding information
National Natural Science Foundation of China, Grant/Award Numbers: 62072077, 62176043; Sichuan Regional Innovation Cooperation Project, Grant/Award Number: 2020YFQ0018; National Key R&D Program of China, Grant/Award Number: 2019YFB1406202

Abstract

Information cascade size prediction is one of the primary challenges for understanding the diffusion of information. Traditional feature-based methods heavily rely on the quality of handcrafted features, requiring extensive domain knowledge and hard to generalize to new domains. Recently, inspired by the success of deep learning in computer vision and natural language processing, researchers have developed neural network-based approaches for tackling this problem. However, existing deep learning-based methods either focused on modeling the temporal characteristics of cascades but ignored the structural information or failed to take the order-scale and position-scale into consideration in modeling structures of information propagation. This paper proposed a novel graph neural network-based model, called MUCas, to learn the latent representations of cascade graphs from a multi-scale perspective, which can make full use of the direction-scale, high-order-scale, position-scale, and dynamic-scale of cascades via a newly designed MUlti-scale Graph Capsule Network (MUG-Caps) and the influence-attention mechanism. Extensive experiments conducted on two real-world data sets demonstrate that our MUCas significantly outperforms the state-of-the-art approaches.

KEYWORDS
capsule network, cascade size prediction, graph neural networks, information cascades, multi-scale features

1 | INTRODUCTION

In the past decade, a large amount of online social platforms have sprung up, such as Twitter, Weibo, Facebook, and so forth. These platforms changed the way people obtain information and brought convenience to the fast diffusion of information.[1,2] Understanding how information is spread has attracted significant attention in both academic world and industry.[3–6] Among various information diffusion tasks, information cascades prediction is one of the primary challenges for understanding the diffusion of information.[7]

Information cascade is formed as information or innovative ideas propagated among users,[8] which has been identified in various settings: for example, retweets in social sites[4,9] and paper citations.[10] The cascade is usually represented as a graph data structure, which means predicting the cascade can be regarded as a graph signal processing problem.[7] Existing works of information cascades prediction can be summarized as two categories, that is, (1) micro-level prediction tasks, which aim at inferring the action status of a specific user,[11,12] and (2) macro-level prediction tasks, which aim at inferring the scale changes of a given cascade.[3–5] In this study, we focus on the macro-level prediction task, specifically, making predictions of the increment size of information cascade after a certain period, which is a challenging but fundamental problem for many real-world applications, for example, viral marketing[6] and misinformation detection,[13–16] and so forth.

As the successes of deep learning methods in many fields, recent studies have developed various neural network-based models to extract diverse features from the cascade graph that can be used for cascade prediction. For example, researchers have leveraged RNNs and attention mechanisms to automatically learn the cascade's temporal characteristics in a sequential learning manner by sampling the cascades as random walks or diffusion paths.[3,4,17,18] These approaches, however, fail to capture topological structure features and the dynamic changes of information diffusion. Later studies[5,19] introduce graph embedding methods to handle the structural diffusion learning problem, and have achieved promising results in cascade prediction. Despite the significant progress made by recent deep learning-based, current approaches still confront several limitations:

(L1) Lack of efficient way to sample cascade graph: Most of the existing studies try to decompose a cascade graph into a bag of nodes[3] or denote it as a set of diffusion paths,[4] such methods either ignore the structural information or fail to capture the time-evolving of the cascade. CasCN[5] breaks down the original cascade graph into a sequence of subgraphs based on timestamps, which may introduce bias and increase computation cost because there are a large number of timestamps in the diffusion process.

(L2) Incomplete structural feature extraction: Structural features are demonstrated as one of the most powerful features in information cascade prediction.[5,19] Existing works not only capture nodes' first-order information but also take the edges' directional information into consideration. However, they still fail to capture long-distance message passing between nodes and nodes' position information in the cascade graph.

(L3) Absence of feature-level fusion: After obtaining different features from the cascade graph, for example, temporal and structural features, most of the current works try to directly concatenate them and then fed them into a fully connected layer to make predictions.[3–5] However, different features play different roles in information cascades prediction, which necessitates a more fine-grained feature fusion that would facilitate predictions.

To overcome the limitations mentioned above, we first define multi-scale information for cascade graph, including (1) direction-scale, representing the propagating direction of the information between nodes; (2) high-order-scale, which is the higher-order interactions between the nodes; (3) position-scale, which means the sequential/position information of each node (i.e., the emerging time of each node in the diffusion); and (4) dynamic-scale, which is the dynamic information captured from the evolving sub-graph sequence. Then, we propose MUlti-scale Cascades model (MUCas)—a novel framework for modeling the information cascades and predicting the increment size of information items. MUCas first employs time interval-aware sub-cascade graph sampling method, which decomposes the observed cascade graph into a sequence of sub-cascade graphs based on *disjoint* time intervals. And then it uses a multi-scale graph capsule network and an influence attention to learn and fuse the multi-scale information to form a unique cascade representation for popularity prediction.

In this study, we make the following contributions:

- Efficient sampling method (L1): We propose a novel cascade sampling method to sample sub-cascade graphs based on disjoint time-intervals. This method can significantly decreases the number of required sub-cascade graphs and eliminate the bias in processing the dynamic-scale in cascade modeling.
- Multi-scale information learning (L2): We design a multi-scale graph network for modeling sub-cascade graphs that can capture direction-scale, high-order-scale, and position-scale features of information diffusion jointly. Simultaneously, we design a neural function to learn the influence-attention between dynamic-scale sub-cascade graphs.
- Hybrid feature aggregation (L3): We propose a capsule-based hybrid aggregation layer, which selectively aggregates the learned multi-scale features in a more fine-grained way, that is, from order-level and node-level to graph-level.
- Comprehensive evaluations: We conducted extensive experiments on two benchmark data sets. The results demonstrate that MUCas can significantly improve the prediction accuracy on cascade size prediction compared with the state-of-the-art baselines.

In the rest of this paper, Section 2 reviews the related work of information cascade prediction and Section 3 formally defines the studied problem. In Section 4, we provide the details of our methodology. Experimental evaluations quantifying the benefits of our approach are discussed in Section 5. We conclude this study in Section 6 with future work remarks.

2 | RELATED WORKS

Information cascades prediction is a general task of interest that is relevant to many downstream social media analysis tasks, such as viral marketing differentiation,[6] influence maximization,[20,21] and misinformation detection,[13–16] and so forth. Existing works on information

cascades prediction generally focus on two levels,[7] that is, macro-level and micro-level. As for the macro-level cascade modeling, researchers aim to learn the holistic and global patterns from the cascades and then infer the size change of cascades, for example, estimate cascade growth[3–5] or forecast outbreaks.[22] In the contrast, at the micro-level cascades, researchers pays more attention to local patterns of social influence, and concerns the behaviors of individual users/items, for example, predicts the activation probability of a specific user.[17,18] In this study, we focus on macro-level tasks, that is, the cascade size prediction, and propose a deep learning-based method—MUCas. Thus, we review existing deep learning-based approaches for macro-level cascade modeling.

With the rapid advancement of deep learning in computer vision and natural language processing, researchers have developed a number of deep models to solve the problem of macro-level information cascades modeling and prediction. The key idea of such deep learning-based models is to automatically extract various diffusion features from the input cascades by leveraging different kinds of neural networks.

DeepCas[3] first demonstrated the effectiveness of deep neural networks in modeling information cascades. It first transformed the cascade graph as a set of node sequences by random walk and then automatically learned the structural features of individual graphs using GRU[23] and the attention mechanism. Li et al. extended DeepCas to DCGT[24] by incorporating content features. DeepHawkes[4] extracted temporal features by modeling diffusion paths via GRU rather than the random walks in DeepCas, and proposed the non-parametric time-decay effect to further improve the prediction performance, which bridged the gap between deep representation learning and the conventional Hawkes process. Gou et al.[22] proposed LSTMIC, which first converted the retweeting time series into several viewpoints, and then employed LSTM and pooling mechanism to extract sequential temporal features for information outbreak prediction. NT-GP[25] extracts node sequences from the user's activity log using the time decay sampling method, and then uses GRU to learn the temporal features from the sampling sequences and predict the target event's future diffusion range. The latest work TempCas[26] introduced a heuristic method for sampling full critical paths that was shown to be more powerful than random walks and diffusion paths. It uses BiGRU with attention pooling for path embedding while modeling the short-term outbreaks and the impact of historical short-term outbreaks with an attention CNN and an LSTM.

Chen et al.[5] proposed the first graph neural networks (GNNs)[27,28]-based model called CasCN. It learns the structural and temporal information from sub-cascade graphs via a combination of graph convolutional network (GCN)[28] and LSTM, which also takes into account the diffusion direction and time-decay effect. Later, some works[29–31] were built upon the CasCN through changing the graph kernel or using different sampling methods. For example, Cascade2Vec[29] improved the convolutional kernel of CasCN with the idea from graph Isomorphism network (GIN)[32] and residual networks.[33] Xu et al. proposed CasGCN,[30] which first represented cascade graph as an in-coming graph and an out-coming graph, and then applied GCN to learn the structural features from both in-coming and out-coming cascade graphs. The temporal features are learned through the normalization of diffusion time. CasSeqGCN[31] assumes that each sampled sub-cascade graph has the same topology but with a different state vector. Huang et al. proposed a graph sequence attention network—GSAN,[34] which captures bidirectional and long dependencies between sub-cascade graphs via a collaboration of graph transformer block and a sequence transformer block. Another work[35]—Coupled GNN learns the cascading effect in information diffusion via coupled GNNs, toward capturing the interpersonal influence and individual user behavior based on the global graph. VaCas[19] first uses

the unsupervised graph wavelet to learn the structural information for cascade graphs, and employs variational autoencoder (VAE)[36] to enhance the cascade representation learning.

Furthermore, some existing works attempt to extract temporal and structural information by performing both micro-level and macro-level tasks concurrently using multi-task learning[37] or reinforcement learning.[12] Also, some works have emerged to solve the general problem inherent in deep cascade learning, such as catastrophic forgetting[38] and long-tail data distribution.[39]

While current deep learning-based approaches achieve significant progress in cascade learning, they still suffer from some limitations such as inefficient sampling, redundant samples, the failure of capturing long-distance message passing, the overlook of nodes' position information in the cascade graph, which are systematically modeled in the proposed MUCas model proposed in this study.

3 | PRELIMINARIES

As mentioned in Section 2, in this paper, we focus on information cascade size prediction. To this end, we cast the cascade size prediction task as a regression problem. In this section, we provide formal definitions of the studied problem and necessary background.

Suppose we have n posts, $P = \{p_i | i \in [1, n]\}$. For each post p_i, with the time elapsed, p_i receives several adoptions, for example, retweets or citations, then the sequence of adoptions forms the information cascade. If we have the concrete propagation path of each adoption, we can generate a cascade graph for p_i, which is formally defined as

Definition Cascade Graph. A cascade graph for a certain post p_i is denoted as $C_i = \{\mathcal{V}_i, \mathcal{E}_i, T_i\}$, where $\mathcal{V}_i$ is a set of nodes, $\mathcal{E}_i \subset \mathcal{V}_i \times \mathcal{V}_i$ is a set of edges and T_i is a set of timestamps. A node $v_* \in \mathcal{V}_i$ can represent a user who tweets or re-tweets the post p_i from some sources (e.g., other users) in social networks or a paper in citation networks. An edge $(v_x, v_y) \in \mathcal{E}_i$ denotes a relationship between v_x and v_y (e.g., re-tweet or citation). $t_* \in T_i$ represents the time when the re-tweeting or citation behavior occurs. Note that the cascade graphs in our paper are dynamic trees.

In our work, we only focus on a portion of the cascade graph C_i. That is, given an observation time t_o, we extract a snapshot of cascade graph C_i before t_o, and the observed snapshot is represented as $C_i(t_o)$. As shown in Figure 1, $C_i(t_o)$ is colored in green background.

As illustrated in Figure 1, we can find that (1) cascade graph is dynamic, in other words, cascade graph is time-evolving, that is, new nodes will join in the diffusion process with time elapsed that leads the cascade size growth. For example, at t_5, v_6 joins in the diffusion process, and make the cascade size increase to 7; (2) the message between two nodes, for example, v_0 and v_1, can be only passed from v_0 to v_1, that is, the message passing in cascade graph is directed; (3) nodes are infected in chronological order, and the sequential information can be regarded as the position of each node in the cascade graph; (4) nodes with high influence will indirectly infect the long-distance nodes, and the long-distance dependency is the higher-order information of each node in a cascade graph, for example, v_0 to v_5.

In this study, we define the aforementioned aspects as the multi-scale information, including (1) dynamic-scale, (2) direction-scale, (3) position-scale, and (4) high-order-scale of cascades.

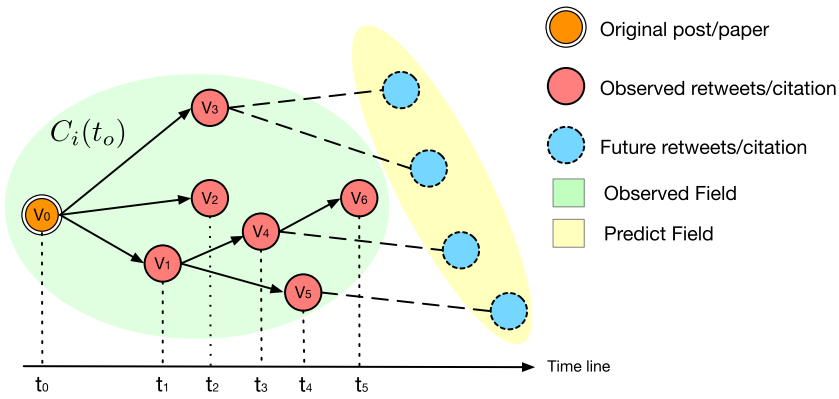


FIGURE 1 A toy example of cascade graph [Color figure can be viewed at wileyonlinelibrary.com]

Definition Cascade Size Prediction. Given the observed cascade graph $C_i(t_o)$ of post p_i, the goal of cascade size prediction is predicting p_i's incremental size ΔS, which is defined as $\Delta S = |\mathcal{V}^{t_e}| - |\mathcal{V}^{t_o}|$, where t_e and t_o are the prediction time and the observation time, respectively; and $|\mathcal{V}^*|$ denotes the number of nodes, in terms of the size of cascade graph.

4 | METHODOLOGY

In this section, we present the proposed MUCas model, as well as its implementation details and computational complexity. Figure 2 illustrates the overall framework and the main components of MUCas, which consists of four major parts: (1) a time interval-aware sampling layer to generate sub-cascade graphs from observed cascade graph; (2) MUG-Caps learns the direction-scale, position-scale and high-order-scale information from sub-cascade graphs; (3) influence attention for dynamic-scale learning; and (4) a prediction layer for cascade size prediction.

4.1 | Time interval-aware sub-cascade sampling

Taking the observed cascade graph $C_i(t_o)$ of a given post p_i as input, the existing works try to decompose the observed cascade graph into a bag of nodes[3] or denote it as a set of diffusion paths.[4] Such methods either ignore both local and global structural information or fail to consider the dynamic information. Recently, some works such as CasCN[5] and VaCas[19] use a *Time-aware* sampling method to decompose $C_i(t_o)$ into a sequence of consecutive sub-cascade graphs based on the diffusion timestamp, which has been proved to be an efficient way to treat the observed cascade graph. However, the *Time-aware* sampling method still faces some challenges: (1) the difference between the fine-grained sub-graphs is trivial, which will introduce biases in dynamic modeling; and (2) too many sub-graphs would significantly increase computation cost. Figure 3A shows a toy example of *Time-aware* sampling method. Compared with the previous time step, each sub-graph only contains one more node (e.g., t_1 vs. t_2). Finally, it would generate m sub-graphs in total, where m is the number of varying time-stamps in the

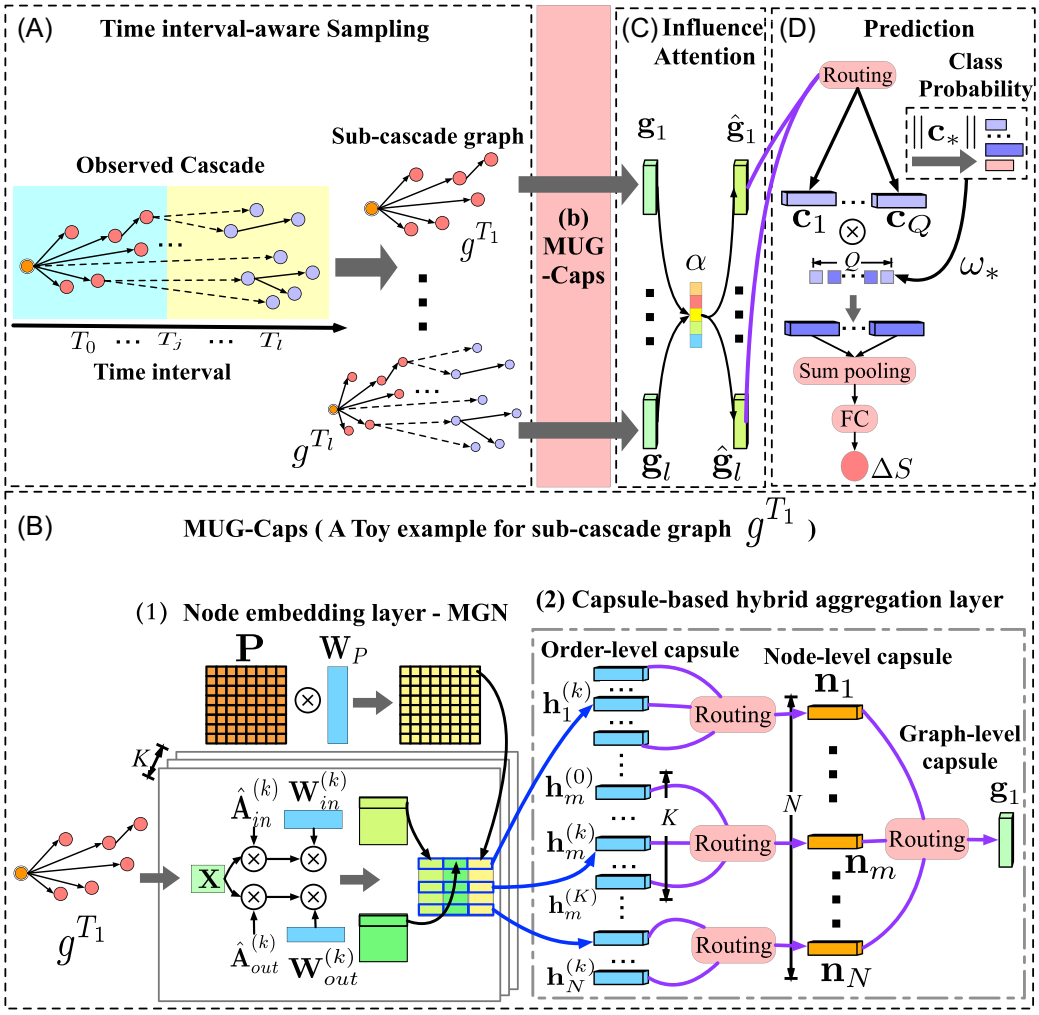


FIGURE 2 Overview of MUCas. (A) A time interval-aware sub-cascade graphs sampling layer; (B) the MUG-Caps layer; (C) the influence attention layer; and (D) the prediction layer [Color figure can be viewed at wileyonlinelibrary.com]

propagation process, resulting a huge number of sub-graphs within a short time. However, the difference between consecutive graphs are too trivial to be distinguished, which may confuse the model to learn discriminative features of information propagation.

To address the aforementioned challenges, we propose a new *Time interval-aware* sampling method, as shown in Figure 3B. This sampling method breaks down the observed cascade graph $C_i(t_o)$ into l discrete sub-cascade graphs $G_i^{T_{iv}} = \{g_i^{T_1}, ..., g_i^{T_j}, ..., g_i^{T_l}, |j \in (1, l)\}$. Specifically, we first split the observation time window t_o into l disjoint time intervals. Then, we sample sub-cascade graphs based on these intervals. Each sub-cascade graph in $G^{T_{iv}}$ is represented by an adjacency matrix. Thus, $G_i^{T_{iv}}$ is further represented as a sequence of adjacency matrices $\mathbf{A}_i^{T_{iv}} = \{\mathbf{A}_i^{T_1}, \mathbf{A}_i^{T_2}, \cdots, \mathbf{A}_i^{T_l}\}$.

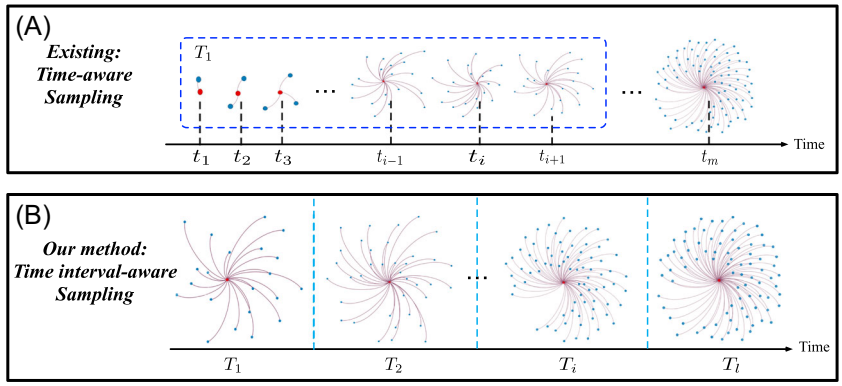


FIGURE 3 Illustration of sampling sub-cascade graph sequence: time-aware sampling (A) versus time interval-aware sampling (B) [Color figure can be viewed at wileyonlinelibrary.com]

Algorithm 1 The algorithm for transforming cascade graph into a fixed-length sub-graph sequence: Time interval-aware sampling

Input: Observed cascade graph $C_i(t_o)$, time window T, and time interval number l.
Output: A fixed-length sub-graph sequence $G_i^{T_{iv}} = \{g_i^{T_1}, \cdots, g_i^{T_l}\}$
1: **for** $n = 1,2,...,l$ **do**
2: **for** Set of nodes $\mathcal{V}_i(t_j)$, set of edges $\mathcal{E}_i(t_j)$ and corresponding timestamp t_j in $C_i(t_o)$**do**
3: Compute the time interval index $m = \lfloor \frac{(t_j - t_0)}{\lceil T/l \rceil} \rfloor + 1$.
4: **if** $m \leq n$ **then**
5: Add $\mathcal{V}_i(t_j)$ and $\mathcal{E}_i(t_j)$ into $g_i^{T_n}$.
6: **end if**
7: **end for**
8: **end for**

Since the proposed sampling rule will transform the observed cascade graph into a fixed number of sub-graphs, it is possible that no new retweet/citation occurs in one of the intervals. To address this issue, we use the sub-graph in front of the empty interval as padding to ensure that the final length of $G^{T_{iv}}$ equals l. Algorithm 1 formalizes the process of the *Time interval-aware* sampling method.

4.2 | Multi-scale cascade representation learning

After generating l discrete sub-cascade graphs, MUCas turns to learn high-level representation of these sub-cascade graphs, which contains the multi-scale information of cascade graphs. Inspired by the recent success of graph neural networks[27,28,40] and capsule network[41] in handling the graph structure data, we propose a MUlti-scale Graph Capsule Network (MUG-Caps) to learn the latent representation for cascade graph from $G^{T_{iv}}$. MUG-Caps is composed of two main parts, that is, (1) *node embedding layer* and (2) *capsule-based hybrid aggregation layer*.

4.2.1 | Node embedding layer

We propose a multi-scale graph network (MGN) as the node embedding module, which learns node representations at the sub-graph level simultaneously from direction-scale, position-scale, and high-order-scale. The implementation of MGN is based on the GCN.[28] Original GCN proposes graph convolution approximations in the spectral domain based on graph Fourier transform, which is computationally efficient and achieves competitive performance in many tasks.[42,43] However, it still faces some limitations in cascade modeling:

(1) GCN focuses on *static* and *undirected* graphs. Nevertheless, the cascade graphs are *dynamic* and *directed* graphs.
(2) GCN updates a node's representation by aggregating its first-order neighbors and itself, failing to capture each node's infected order, that is, node's position information.
(3) GCN aggregates the high-order information for a node through stacking multiply graph convolutional layers. As demonstrated by many later improved works,[40,44,45] deeper GCN could not improve the performance and even performs worse in graph representation learning.

Our MGN addresses these limitations through revising the convolution kernel of GCN, which is defined as

$$\mathbf{H} = g_\theta * \mathbf{X} = \sigma\left[\Big\|_{k\in\mathcal{O}}, \Big\|_{\phi\in\{in,out\}} \left(\hat{\mathbf{A}}_\phi^{(k)}\mathbf{X}\mathbf{W}_\phi^{(k)}\right), \mathbf{P}\mathbf{W}_P\right], \tag{1}$$

where $\|_{k\in\mathcal{O}}$ and $\|_{\phi\in\{in,out\}}$ represent the order-level concatenation and direction-level concatenation, respectively; $[\cdot]$ is a tiling concatenate operation; σ is an element-wise activation such as ReLU; $\hat{\mathbf{A}}_\phi^{(k)}$ denotes the normalized adjacency matrix $\hat{\mathbf{A}}_\phi \in \mathbb{R}^{N\times N}$ multiplied by itself k times, specifically, $\hat{\mathbf{A}}_\phi^{(0)} = \mathbf{I}$ is an identity matrix; N is the number of nodes in current sub-cascade graph; $\mathbf{X} \in \mathbb{R}^{N\times F}$ is the input graph signal – F is the dimension number; and $\mathcal{O}$ is a set of integer adjacency powers – the value of $\mathcal{O}$ is from 0 to the max-order K of the current sub-cascade graph. $\phi \in \{in, out\}$ represents the in- and out-directions of the adjacency matrix, respectively. $\mathbf{A} = \mathbf{A}_{in} = (\mathbf{A}_{out})^T$. The asymmetric normalized adjacency matrix $\hat{A}_\phi$ of each direction can be calculated as

$$\begin{aligned}\hat{\mathbf{A}}_\phi &= (\bar{\mathbf{D}}_\phi)^{-1}\bar{\mathbf{A}}_\phi,\\ \bar{\mathbf{A}}_\phi &= \mathbf{A}_\phi + \mathbf{I}_N,\end{aligned} \tag{2}$$

where $\mathbf{I}_N$ is the identity matrix, and $(\bar{\mathbf{D}}_\phi)_{ii} = \sum_j (\bar{\mathbf{A}}_\phi)_{ij}$ is the diagonal degree matrix.

In MGN, the initial $\mathbf{X} = \mathbf{A}\cdot\mathbf{P} \in \mathbb{R}^{N\times F_p}$ is a position embedding matrix for current sub-cascade graph, and F_p is an adjustable dimension. Specifically, we initialize the position embedding matrix $\mathbf{P} = \{\mathbf{p}_1, ..., \mathbf{p}_u, ..., \mathbf{p}_N\}$ through positional encoding $\mathbf{PE}(u)$[46] as

$$\begin{aligned}\mathbf{PE}(u)_{2d} &= \sin\left(u/10{,}000^{2d/d_p}\right),\\ \mathbf{PE}(u)_{2d+1} &= \cos\left(u/10{,}000^{2d/d_p}\right),\end{aligned} \tag{3}$$

where $1 \le d \le F_p/2$ denotes the dimension index in $\mathbf{p}_u$. The details of this formula are referred to.[46]

In Equation (1), $\mathbf{W}_\phi^{(k)} \in \mathbb{R}^{F \times F_d}$ is the weight matrix for each direction on different order, and $\mathbf{W}_P \in \mathbb{R}^{d_p \times F_p}$ is another weight matrix used to transform position embeddings. The output node embedding matrix is denoted as $\mathbf{H} \in \mathbb{R}^{N \times K \times (2F_d + F_p)}$ and $\mathbf{H} = \{\mathbf{h}_1, \cdots, \mathbf{h}_m, \cdots, \mathbf{h}_N\}$, respectively. When implementing MGN, there is no need to calculate $\hat{\mathbf{A}}_\phi^k$ for each order. Instead, we calculate $\hat{\mathbf{A}}_\phi^k \mathbf{X}$ via right-to-left multiplication. For example, when $k = 2$, $\hat{\mathbf{A}}_\phi^2 \mathbf{X}$ is calculated as $\hat{\mathbf{A}}_\phi(\hat{\mathbf{A}}_\phi(\mathbf{IX}))$, where $\mathbf{I}$ is the identity matrix. MGN can be regarded as a single layer using multiple times during actual training. The calculation of MGN is outlined in Algorithm 2.

The rationale behind MGN: MGN handles the *direction-scale* of cascade through modeling the incoming and outgoing relations of the cascades, that is, $\hat{\mathbf{A}}_\phi^{(k)} \mathbf{X} \mathbf{W}_\phi^{(k)}$ in Equation (1). Moreover, due to the directed graph is asymmetric, we use asymmetric normalization $\hat{\mathbf{A}}_\phi = (\hat{\mathbf{D}}_\phi)^{-1} \bar{\mathbf{A}}_\phi$ to replace the symmetric normalization $\hat{\mathbf{D}}^{-\frac{1}{2}} \bar{\mathbf{A}} \hat{\mathbf{D}}^{-\frac{1}{2}}$ used in vanilla GCN. MGN utilizes the adjacency matrix's multiple powers to mix the feature representations of higher-order neighbors in one graph convolutional layer, which is used to handle the *high-order-scale*. In our implementation, the sub-cascade graph is a tree, and the value of $\mathbf{A}_{ij}^{(k)}$ can be either 0 or 1. When $\mathbf{A}_{ij}^{(k)} = 1$, there is a path between v_i to v_j. For arbitrary power p and q, $\mathbf{A}_{ij}^p \mathbf{A}_{ij}^q = 0$ will always be held, which eliminates the problem of layer output—imposing the lower-order information on higher-order relations and increase the feature correlations.[47] As for the *position-scale*, MGN adds position embeddings to the convolution kernel, which enriches each node feature with its corresponding position information.

Algorithm 2 Calculation of multi-scale graph network

Input: Feature matrix $\mathbf{X}$, normalized adjacency matrix $\hat{\mathbf{A}}_\phi$ for both in- and out-directions, a set of order powers $\mathcal{O}$ and its max-value $K = \max(\mathcal{O})$, and positional embedding matrix $\mathbf{P}$.

Parameters: $\{\mathbf{W}_\phi^{(k)}\}_{k \in \mathcal{O}}$ and $\mathbf{W}_P$.

$\hat{\mathbf{P}} \coloneqq \mathbf{P}\mathbf{W}_P$
$\mathbf{B}_{in}, \mathbf{B}_{out} \coloneqq \mathbf{X}$
for k in $\mathcal{O}$ **do**
  if $k = 0$ **then**
    $\mathbf{B}_{in} \coloneqq \mathbf{I}\mathbf{B}_{in}$
    $\mathbf{B}_{out} \coloneqq \mathbf{I}\mathbf{B}_{out}$
  else
    $\mathbf{B}_{in} \coloneqq \hat{\mathbf{A}}_{in}\mathbf{B}_{in}$
    $\mathbf{B}_{out} \coloneqq \hat{\mathbf{A}}_{out}\mathbf{B}_{out}$
  end if
  $\mathbf{H}^{(k)} \coloneqq \mathrm{CONCAT}\left(\mathbf{B}_{in}\mathbf{W}_{in}^{(k)}, \mathbf{B}_{out}\mathbf{W}_{out}^{(k)}, \hat{\mathbf{P}}\right)$
end if
$\mathbf{H} \coloneqq \mathrm{CONCAT}(\mathbf{H}^{(0)}, \cdots, \mathbf{H}^{(k)}, \cdots, \mathbf{H}^{(K)})$
return $\sigma(\mathbf{H})$

4.2.2 | Capsule-based hybrid aggregation layer

From the node embedding layer, we obtain a set of node embedding matrix for each sub-cascade graph, denoted as $\mathbf{S} = \{\mathbf{H}_1, \mathbf{H}_2, \cdots, \mathbf{H}_l\}$, where $\mathbf{S} \in \mathbb{R}^{l \times N \times K \times (2F_d + F_p)}$. Inspired by the work of capsule networks,[41,48–51] we design a capsule-based hybrid aggregation layer to aggregate the learned node features from order-level, node-level, and graph-level through dynamic routing, respectively.

Algorithm 3 Dynamic routing mechanism.

Input: Lower-level capsules $\mathbf{U}$, iteration number τ.
Output: Upper-level capsules $\mathbf{S}$
1: for all lower-level capsules i: $\hat{\mathbf{v}}_{j|i} = \mathbf{W}_{ij}\mathbf{u}_i$
2: $\mathbf{b}_{ij} \leftarrow 0$
3: **for** τ iterations **do**
4: for all lower-level capsules i: $c_{ij} \leftarrow \text{softmax}(\mathbf{b}_{ij})$
5: for all upper-level capsules j: $\mathbf{s}_j \leftarrow \sum_i c_{ij}\hat{\mathbf{v}}_{j|i} + \mathbf{b}_j$
6: for all upper-level capsules j: $\tilde{\mathbf{s}}_j \leftarrow \text{Squash}(\mathbf{s}_j)$
7: for all lower-level capsules i to all upper-level capsules j: $\mathbf{b}_{ij} \leftarrow \mathbf{b}_{ij} + \hat{\mathbf{v}}_{j|i} \cdot \tilde{\mathbf{s}}_j$
8: **end for**

The general procedure of dynamic routing is shown in Algorithm 3: (1) Lower-level capsules $\mathbf{U} \in \mathbb{R}^{|\mathbf{U}| \times F_U}$ are linearly transformed through shared matrix $\mathbf{W} \in \mathbb{R}^{|\mathbf{U}| \times |\mathbf{S}| \times F_U \times F_S}$, where $|\mathbf{U}|$ and F_U are the number of lower-level capsules and the dimensions, respectively. Here we introduce $\mathbf{W}$ that not only guarantees the feature representation ability of the center vector after clustering, but also being able to identify the order of input features. The result of this step is a set of votes $\hat{\mathbf{V}} \in \mathbb{R}^{|\mathbf{U}| \times |\mathbf{S}| \times F_S}$ (cf. line 1 in Algorithm 3), where $|\mathbf{S}|$ and F_S are the number of upper-level capsule and the dimensions, respectively. (2) Upper-level capsules $\mathbf{S} \in \mathbb{R}^{|\mathbf{S}| \times F_S}$ are computed based on the votes via line 3–7 in Algorithm 3, where $c_{ij} \in \mathbb{R}^{|\mathbf{U}| \times |\mathbf{S}| \times 1}$ is the coupling co-efficiency that helps weight the votes, and the nonlinear "squash" function is denoted as $\text{Squash}(x) = \frac{\| x \|^2}{0.5 + \| x \|^2} \frac{x}{\| x \|}$.

We add biases to the calculation of $\mathbf{s}_j$ at line 5, which can solve a critical problem in capsule networks—indistinguishableness between the positive inputs and negative inputs.[52]

In the MGN, we defined three levels of capsules, that is, order-level capsule, node-level capsule, and graph-level capsule, whose specific definitions are as follows:

- Order-level capsule: It is represented as $\mathbf{h}_m \in \mathbb{R}^{K \times (2F_d + F_p)}$, focusing on specific node embedding in the sub-cascade graph. It aims to aggregate the higher-order information for each node into one node-level capsule $\mathbf{n}_m \in \mathbb{R}^{1 \times F_n}$, where F_n is the dimension of node-level capsule.
- Node-level capsule: As for a specific sub-cascade graph g^{T_j}, it has a set of node-level capsules $\mathbf{N}_j = \{\mathbf{n}_1, \cdots, \mathbf{n}_m, \cdots, \mathbf{n}_N\}$, $\mathbf{N}_j \in \mathbb{R}^{N \times F_n}$, used $\mathbf{N}_j$ to generate the graph-level capsule $\mathbf{g}_j \in \mathbb{R}^{1 \times F_g}$ via dynamic routing, where F_g is the dimension of graph-level capsule.

Graph-level capsule: We have a set of graph-level capsules $\mathbf{G}$, each of which corresponds to a specific sub-cascade graph, that is, $\mathbf{G} = \{\mathbf{g}_1, \cdots, \mathbf{g}_j, \cdots, \mathbf{g}_l\}$, $\mathbf{G} \in \mathbb{R}^{l \times F_g}$. Note that each graph-level capsule contains the properties of the cascade from different time intervals.

How does MUG-Caps work? Above we presented the details of the two components of MUG-Caps, that is, MGN and the capsule-based hybrid aggregation layer. Now we turn to explain how the two components collaborate. MUG-Caps, as shown in Figure 2, takes a subgraph as input, which is first fed into an MGN layer to learn the embedding for each node that contains node direction-scale information, higher-order-scale information, and position-scale information as detailed in Section 3. After this layer, each node was represented as K vectors, with each vector matched to a different order level, referred to as the order-level capsules. Next, these order-level capsules are fed into the hybrid aggregation layer. Then dynamic routing is employed to aggregate order-level capsules to form a node-level capsule for each node. Finally, the dynamic routing aggregates these node-level capsules to build a graph-level capsule for the subgraph. The concrete calculation of MUG-Caps is shown in steps 3 to 11 in Algorithm 4.

4.3 | Sub-graph level influence attention

Previous works have demonstrated that user influence will decay significantly with time.[4,5] In our work, we aim to learn such influence changes (dynamic-scale) at the sub-graph level, that is, we assume that the sub-cascade graph's influence decays as the interval index increases. Inspired by self-attention mechanism,[53] we employ a neural function to learn the influence attention. First, we represent the time-interval as a one-hot vector $\mathbf{t}^j \in \mathbb{R}^l$, and then map $\mathbf{t}^j$ to $\boldsymbol{\lambda}_j$ through a fully-connected layer with sigmoid function. Here the $\boldsymbol{\lambda}_j$ is used to describe the time decay effect.

$$\boldsymbol{\lambda}_j = \text{sigmoid}(\mathbf{W}_t \mathbf{t}^j + \mathbf{b}_t), \tag{4}$$

where $\mathbf{W}_t \in \mathbb{R}^{F_g \times l}$ and $\mathbf{b}_t \in \mathbb{R}^{F_g}$. According to the time decay effect vector $\boldsymbol{\lambda}_j$ and graph-level capsules $\mathbf{G}$, we define the influence attention as

$$\alpha_j = \frac{\exp(\langle \mathbf{w}, \boldsymbol{\lambda}_j \odot \mathbf{g}_j \rangle)}{\sum_{i=1}^{l} \exp(\langle \mathbf{w}, \boldsymbol{\lambda}_i \odot \mathbf{g}_i \rangle)}, \tag{5}$$

where $\mathbf{w} \in \mathbb{R}^{F_g}$. Given the influence attention α_j, we calculate the graph-level capsule as

$$\hat{\mathbf{g}}_j = \alpha_j \mathbf{g}_j. \tag{6}$$

4.4 | Information cascade prediction

Though our work focus on information popularity prediction, unlike existing works, we add an auxiliary task—an extra classification task, that is, whether a cascade would go viral, as a supplementary for the cascade size prediction. That is, we predict whether a cascade can break out a certain threshold value. This step is also implemented using dynamic routing over

$\hat{\mathbf{G}} = \{\hat{\mathbf{g}}_j | j \in [1, l]\}$ to generate class capsules $\mathbf{C} \in \mathbb{R}^{Q \times F_c}$, where Q is the number of class, and F_c is the dimension of class capsules. The norm of class capsule $\|\mathbf{c}_q\|$ represents the probability belonging to class q. And, we use a margin loss to calculate the classification loss:

$$\ell_1 = \sum_q \{\mu_q \max(0, m^+ - \|\mathbf{c}_q\|)^2 + \xi(1 - \mu_q)\max(0, \|\mathbf{c}_q\| - m^-)^2\}, \tag{7}$$

where $m^+ = 0.9$, $m^- = 0.1$, and $\mu_q = 1$ iff the cascade belongs to class q.[41,48,52] Here ξ is used to stop initial learning from reducing the length of all class capsules, especially when Q is large.

Subsequently, we use a weighted sum operation on $\mathbf{C}$ to obtain the representation for a cascade $\hat{\mathbf{C}}$. The weight is calculated through $\|\mathbf{c}_*\|$:

$$\begin{aligned} \omega_q &= \frac{\exp(\|\mathbf{c}_q\|)}{\sum_{*=1}^{Q} \exp(\|\mathbf{c}_*\|)}, \\ \hat{\mathbf{C}} &= \sum_q \omega_q \mathbf{c}_q. \end{aligned} \tag{8}$$

We use a fully-connected layer to predict the increment size $\Delta S = \mathrm{FC}(\hat{\mathbf{C}})$. The loss function is

$$\ell_2 = (\log \Delta S - \log \Delta \widetilde{S})^2, \tag{9}$$

where $\Delta \widetilde{S}$ is the ground truth. Finally, the overall loss function for a batch is

$$\mathcal{L} = \frac{1}{B} \sum_{i=1}^{B} \left(\beta \ell_1^i + (1 - \beta)\ell_2^i\right), \tag{10}$$

where B is the number of cascades in a batch, and β is used to balance the ℓ_1 and ℓ_2 losses. Algorithm 4 illustrates the training process of MUCas.

Algorithm 4 Learning with *MUCas*.

Input: Time-interval aware sub-cascade graphs for n cascade graphs $\mathbf{A}^{T_{iv}} = \left\{\mathbf{A}_1^{T_{iv}}, \cdots, \mathbf{A}_i^{T_{iv}}, \cdots, \mathbf{A}_n^{T_{iv}},\right\}$; Max order K; Batch size b.

Output: Incremental size $\Delta S = \{\Delta S_1, \cdots\}$ of cascades.

1: **repeat**
2: $b = 1, 2, \ldots$
3: **for** adjacency matrix sequence $\mathbf{A}_i^{T_{iv}}$ in batch b**do**
4: **for** $\mathbf{A}_i^{T_j} \in \mathbf{A}_i^{T_{iv}}$**do**
5: Compute the node embeddings $\mathbf{H}_j$ for j^{th} sub-cascade graph according to Eq. (1).
6: **for** node embedding $\mathbf{h}_m \in \mathbf{H}_j$**do**
7: Use $\mathbf{h}_m$ to compute the node-level capsule $\mathbf{n}_m$ according to Algorithm 3.
8: **end for**
9: $\mathbf{N}_j \leftarrow \{\mathbf{n}_1, \cdots, \mathbf{n}_N\}$

10: Use $\mathbf{N}_j$ to generate the graph-level capsule $\mathbf{g}_j$ according to Algorithm 3.
11: **end for**
12: $\mathbf{G}_i \leftarrow \{\mathbf{g}_1, \cdots, \mathbf{g}_l\}$
13: Compute influence attention α_j according to Eq. (4) and (5).
14: Compute newly graph-level capsules $\hat{\mathbf{G}}_i$ according to Eq. (6).
15: /*Extra classification task*/
16: Use $\hat{\mathbf{G}}_i$ to compute class capsules $\mathbf{C}_i$ according to Algorithm 3.
17: /*Popularity prediction task*/
18: Calculate cascade representation $\mathbf{C}_i^{'}$ according to Eq. (8).
19: Feed $\mathbf{C}_i^{'}$ into fully-connected layer to compute incremental size ΔS_i of cascade.
20: Use Adaptive moment estimation (Adam) to optimize the objective function in Eq. (10) and update all the parameters.
21: **end for**
22: **until** convergence;

4.5 | Complexity analysis

We finalize this section with a discussion of the computational complexity of the main component *MUG-Caps* in MUCas, that is, the cost of (1) node embedding layer, and (2) capsule-based hybrid aggregation layer.

4.5.1 | Node embedding layer

We used MGN to learn node embeddings from the sub-cascade graph as shown in Equation (1). As for *direction-scale*, MGN models the incoming and outgoing relations of the cascades via $\hat{\mathbf{A}}_{\phi}^{(k)}\mathbf{X}\mathbf{W}_{\phi}^{(k)}$. Recall that the dimensions of $\mathbf{X}$, $\mathbf{W}_{\phi}^{(k)}$, and $\mathbf{W}_p$ are F, F_d, and F_p, respectively. Besides, the max-order is K, and the normalized adjacency matrix $\hat{\mathbf{A}}_{\phi}$ in our implementation is a sparse matrix with $|\mathcal{E}|$ nonzero elements—$|\mathcal{E}|$ is the number of edges in current graph. In addition, the number of nodes is denoted as N. According to existing works,[28,40] for a single direction at each order, the calculation is conducted via sparse-dense matrix multiplications, and the computational complexity is $\mathcal{O}(|\mathcal{E}| \times F \times F_d)$. Taking the high-order-scale into consideration, the computational complexity is then $\mathcal{O}(K \times |\mathcal{E}| \times F \times F_d)$. As for *positional-scale*, the calculation of $\mathbf{W}_p\mathbf{P}$ is completed via matrix multiplication, which requires $\mathcal{O}(N \times d_p \times F_p)$ computations. Therefore, the total computational time cost of evaluating Equation (1) is then $\mathcal{O}(2 \times K \times |\mathcal{E}| \times F \times F_d + N \times d_p \times F_p)$.

4.5.2 | Capsule-based hybrid aggregation layer

This layer is implemented by executing dynamic routing between different levels capsules. Specifically, we adopt a dynamic routing mechanism for τ iterations over $|\mathbf{U}|$ lower-level capsules and generate $|\mathbf{S}|$ upper-level capsules. This learning process requires $\mathcal{O}(\tau \times |\mathbf{U}| \times |\mathbf{S}|)$ computations.[54] From the MGN, we get the order-level capsule $\mathbf{H} = \{\mathbf{h}_1, \cdots, \mathbf{h}_m, \cdots, \mathbf{h}_N | \mathbf{h}_m \in \mathbb{R}^{K \times (2F_d + F_p)}\}$. For each node m, it generates the node-level capsule from order-level capsules, whose computational complexity is $\mathcal{O}(\tau \times K \times 1)$. Because there are N nodes, the total time of generating node-level

capsules for all nodes is $\mathcal{O}(N \times \tau \times K \times 1)$. Similarly, generating the graph-level capsule from node-level capsules can be done within $\mathcal{O}(\tau \times N \times 1)$ time. The overall computational time is therefore $\mathcal{O}(N \times \tau \times K \times 1 + \tau \times N \times 1)$.

5 | EXPERIMENTS

In this section, we compare the performance of our proposed model *MUCas* with the state-of-the-art approaches, and several variants of *MUCas*, on information popularity prediction. In particular, we provide the quantitative results to answer the following research questions:

- RQ1: How does MUCas perform on cascade size prediction compared with the state-of-the-art baselines?
- RQ2: How do different scales of information modeled in MUCas contribute to the overall performance?
- RQ3: How do the key hyper-parameters affect the performance of MUCas?

5.1 | Data sets

We evaluate the effectiveness and generalizability of *MUCas* on two scenarios. The first one is to predict the size of retweet cascades in Sina Weibo and the second one is to forecast the citation count of papers in citation data set APS. The statistics of the data sets used in this study has shown in Table 1.

5.1.1 | Sina Weibo

Weibo data set is released by Cao et al.,[4] and collected from Sina Weibo (http://www.weibo.com)—one of the most popular microblog platform in China. This data set collects the posts generated on June 1, 2016, and tracks all retweets for each post within the next 24 h. It consists of 119, 313 posts in total. Figure 4A plots the distribution of the cascade size (the number of retweets of each post), which, obviously, follows an power-law distribution and reflects the Pareto principle (80/20 rule). Figure 5A shows the distribution of depth over all cascades, which roughly follows an exponential distribution, indicating that the majority of cascades have a shallow depth, that is, most of them are less than 5. The depth of a cascade is the length of the longest path, which also equals to the max-order of the cascade. Due to the effect of diurnal rhythm in Weibo,[4] in our experiments, the cascades with the publication time before 8 a.m. and

TABLE 1 Statistics of data sets

Data sets	# Ori Cascades	# Nodes	# Edges	Avg. popularity	0.5 h/3 years			1 h/5 years		
					Train	Val	Test	Train	Val	Test
Weibo	119,313	5,918,473	12,204,245	173	19,472	4173	4172	24,636	5279	5278
APS	636,294	636,294	3,425,508	13	19,124	4098	4097	33,408	7159	7158

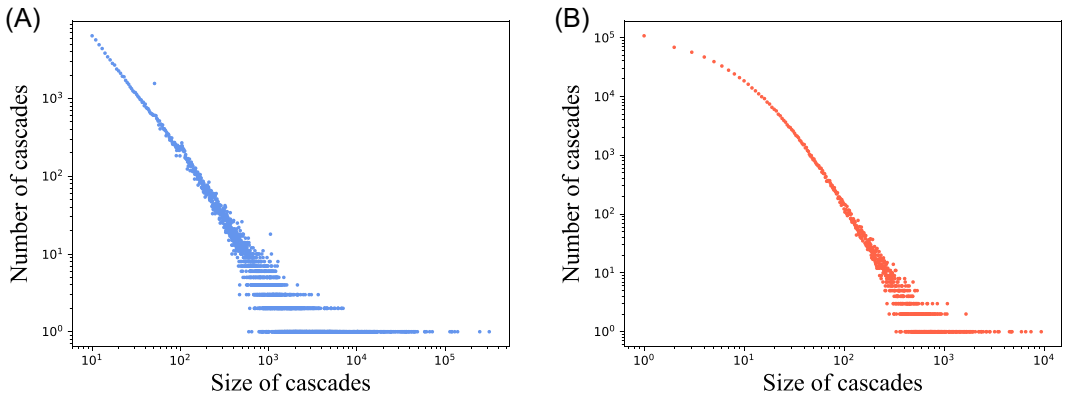


FIGURE 4 Cascade size distributions of Weibo (A), and APS (B) data sets in log-log scales [Color figure can be viewed at wileyonlinelibrary.com]

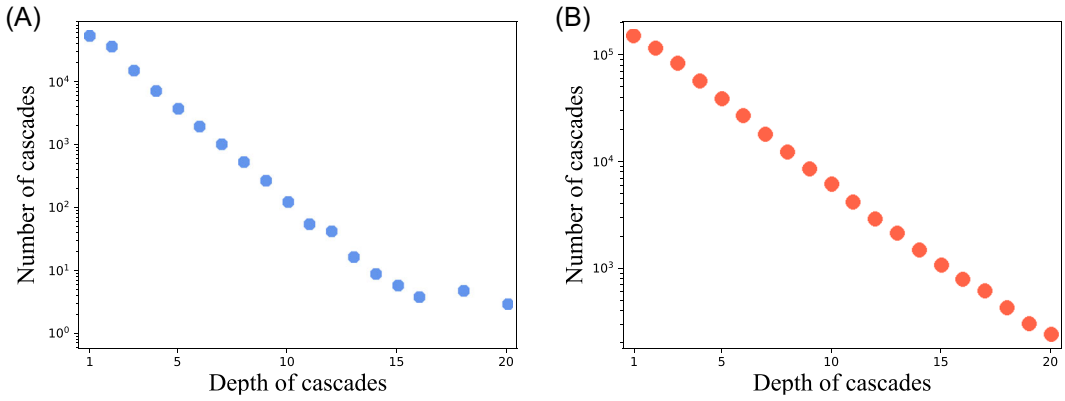


FIGURE 5 Distribution of depth (max-order) of cascades. (A) Weibo data set and (B) APS data set [Color figure can be viewed at wileyonlinelibrary.com]

after 6 p.m. were filtered out, leaving each post at least 6 h to obtain retweets. As shown in Figure 6A, on average, a message receives about 70% retweets within 5 h.

5.1.2 | APS

APS (http://journals.aps.org) APS is provided by American Physical Society (APS), which consists of pairs of APS articles that cite each other for the corpus of Physical Review Letters, Physical Review, and Reviews of Modern Physics from 1893 to 2018. The papers from 1893 to 1997 are selected as observations so that each of the papers is allowed to develop for at least 20 years. In the citation scenario, the size of a cascade is the citation count. Figure 4B shows the distribution of cascade size in the APS data set, which exhibits a power-law distribution. Figure 5B shows the distribution of the depth over all cascades in APS, which has a similar trend with Weibo. As shown in Figure 6B, on average, the citation reaches around 50% of the final size within 5 years.

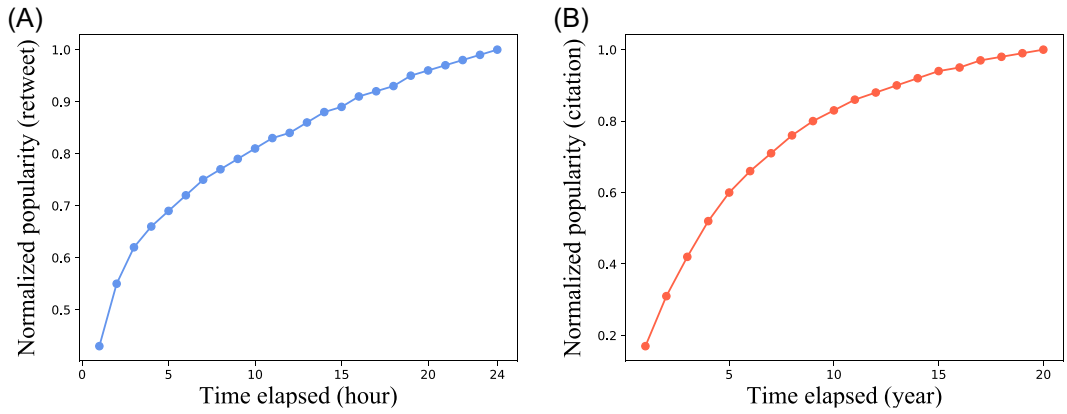


FIGURE 6 Size distribution of cascades: retweets (citations) versus time. (A) Weibo data set and (B) APS data set [Color figure can be viewed at wileyonlinelibrary.com]

5.1.3 | Settings

In our experiments, we use the same data set settings as in previous works.[4,19] The observation time window in Weibo data is set to $T = 0.5$ h and $T = 1$ h. For the observation window of the APS data set, we choose $T = 3$ and 5 years. For both Weibo and APS data sets, we filter out cascades whose observed size $S_{obs} < 10$. And for those cascades whose $S_{obs} > 100$, we only track the first 100 retweets. In addition, we randomly split each data set into a training set (70%), a validation set (15%), and a testing set (15%) following existing works.[4,19]

5.2 | Baselines

To validate MUCas's performance in cascade prediction, we select following state-of-the-art baselines for comparison:

- Handcrafted Feature-based: Cheng et al.[8] defined five classes of features, including content features, original poster, resharer features, structual and temporal features. Specifically, in this study, we extract the cumulative popularity series, time between original and first participant, mean time between the first half and the second half of participants, number of leaf nodes, mean node degree, mean and max length of sequences. Then, we feed these extracted features into a linear regression model and a fully-connected layer to predict the increment size, and denote these two independent methods as *Feature-Linear* and *Feature-Deep*, respectively.
- DeepCas[3]: DeepCas uses random walks to represent a cascade graph and predict the increment size of cascades through bidirectional GRU[23] and an attention mechanism. It only utilizes cascade structure and node identities in the prediction.
- DeepHawkes[4]: DeepHawkes combines both the end-to-end deep learning and Hawkes process for cascade size prediction, which bridges the gap between prediction and the mechanism ruling information propagation. Specifically, it uses deep learning methods to solve

the three key aspects in Hawkes process, that is, influence of users, self-exciting mechanism, and time decay effect.

- CasCN[5]: CasCN is the first graph convolution network (GCN)-based framework exploiting both temporal and structural information for cascade size prediction. It decomposes a cascade graph into a sequence of sub-cascade graphs based on propagation time, while learning the local structure and its evolving process of cascade structure through the combination of graph convolutions and LSTM.
- VaCas[19]: VaCas is the first Bayesian learning-based approach that uses pre-trained node embeddings of the cascade as input and leverages a hierarchical variational information diffusion model to learn the posterior of cascade distribution with variational inference.
- Cascade2vec[29]: Cascade2vec is an improvement of CasCN, which proposes a new graph convolutional kernal—graph perception network (GPN) to replace the original GCN in CasCN. It also introduces the attention mechanism to learn different importance of each sub-graph.

5.3 | Evaluation metric

Following previous studies,[3,4,17] we use mean square logarithmic error (MSLE) and symmetric mean absolute percentage error (SMAPE) for prediction performance evaluation. In addition, we also report the coefficient of determination (R^2) of different models. The formulation of all evaluation metric is defined as

$$\begin{aligned} \text{MSLE} &= \frac{1}{n}\sum_{i=1}^{n}(\log \Delta S_i - \log \Delta \widetilde{S}_i)^2, \\ \text{SMAPE} &= \frac{1}{n}\sum_{i=1}^{n}\frac{|\log \Delta S_i - \log \Delta \widetilde{S}_i|}{(|\log \Delta S_i| + |\log \Delta \widetilde{S}_i|)/2}, \\ R^2 &= 1 - \frac{\sum_{i=1}^{n}(\log \Delta S_i - \log \Delta \widetilde{S}_i)^2}{\sum_{i=1}^{n}\left(\log \Delta \widetilde{S}_i - \frac{1}{n}\sum_{i=1}^{n}\log \Delta \widetilde{S}_i\right)^2}, \end{aligned} \tag{11}$$

where n is the total number of posts, ΔS_i is the predicted incremental size for post p_i, and $\Delta \widetilde{S}_i$ is the ground truth. Note that, the value of MSLE and SMAPE the smaller the better, in contrast, the value of R^2 the bigger the better.

5.4 | Experimental settings and parameter tuning

5.4.1 | Parameter settings

Models mentioned above are optimized to the best performance which involves several key hyper-parameters. The L_2 coefficient of Feature-Linear is chosen from $10^{\{0,-1,-2\cdots,-8\}}$. The node embedding size for DeepCas, DeepHawkes, CasCN, and Cascade2vec is set to 50. The hidden layer of each GRU has 32 units, and the hidden dimensions of the two-layer fully-connected layers for all deep learning-based methods are 32 and 16, respectively. The learning rate for node embeddings in DeepCas and DeepHawkes is 5×10^{-4} and the learning rate for other

methods are 5×10^{-3}. The batch size is set as 64. All other hyper-parameters are set to the same values as used in the original papers.

As for our MUCas, the basic parameters (e.g., the learning rate is 5×10^{-4} and batch size is 64, etc.) are the same as above deep learning-based approaches, except that the max-order K and iteration number τ are chosen from 1 to 5. The embedding size of positional embedding is chosen from {30, 50, 100, 150, 200}, the hidden size for MGN is 60, and the hidden size for node-level capsule, graph-level capsule and the class capsule is 30, 8 and 16, respectively. In addition, the number of time intervals is set to 6. All methods, including ours, are tuned to the best performance with early stopping when validation errors has not declined for 10 consecutive epochs.

5.4.2 | Experimental environment

The experiments are conducted on a sever with Intel E5-2680 v4 2.40 GHz, one NVIDIA GeForce GTX 3090, and 256 GB memory.

5.5 | Performance comparison (RQ1)

The overall performance of MUCas as well as the state-of-the-art baselines are shown in Tables 2 and 3, from which we have the following important observations.

(O1) The performance of MUCas outperforms the baselines by a large margin, for example, as for the MSLE, it reduces the prediction error up to 19.53%, 20.22%, 9.63%, and 4.51% comparing to the best baseline—VaCas, when T is set to 0.5, 1 h and 3, 5 years on Weibo and

TABLE 2 Performance comparison between baselines and MUCas on Weibo data sets

	Weibo					
	0.5 h			1 h		
Model	MSLE	SMAPE	R^2	MSLE	SMAPE	R^2
Baselines						
Feature-Linear	2.959	0.331	0.351	2.710	0.356	0.461
Feature-Deep	2.815	0.311	0.379	2.646	0.353	0.465
DeepCas	2.914	0.330	0.352	2.747	0.358	0.459
DeepHawkes	2.891	0.321	0.379	2.632	0.352	0.468
CasCN	2.804	0.311	0.381	2.601	0.350	0.468
Cascade2vec	2.752	0.308	0.384	2.589	0.348	0.479
VaCas	2.586	0.291	0.504	2.359	0.333	0.518
Ours						
MUCas	**2.081***	**0.271***	**0.621***	**1.882***	**0.308***	**0.647***
(Improvements)	19.53%	6.87%	23.21%	20.22%	7.51%	24.90%

Note: A paired t test is performed. The best method is shown in bold, and the second best is shown as underlined.

*Statistical significance $p < 0.001$ as compared with the best baseline method.

TABLE 3 Performance comparison between baselines and MUCas on APS data sets

	APS					
	3 Years			5 Years		
Model	**MSLE**	**SMAPE**	R^2	**MSLE**	**SMAPE**	R^2
Baselines						
Feature-Linear	2.100	0.289	0.126	2.087	0.358	0.311
Feature-Deep	1.996	0.358	0.221	1.874	0.352	0.322
DeepCas	2.033	0.361	0.213	1.944	0.365	0.318
DeepHawkes	1.831	0.344	0.241	1.588	0.337	0.363
CasCN	1.818	0.274	0.244	1.574	0.337	0.367
Cascade2vec	1.783	0.272	0.258	1.560	0.336	0.373
VaCas	1.723	0.268	0.283	1.507	0.335	0.394
Ours						
MUCas	**1.557***	**0.263***	**0.355***	**1.439***	**0.333***	**0.426***
(Improvements)	9.63%	1.87%	25.44%	4.51%	0.59%	8.12%

Note: A paired t test is performed. The best method is shown in bold, and the second best is shown as underlined.
*Statistical significance $p < 0.001$ as compared with the best baseline method.

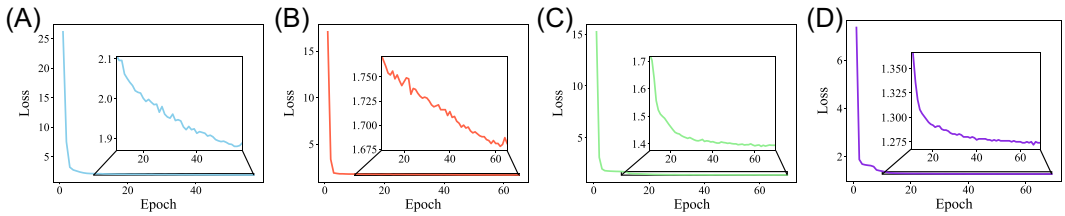


FIGURE 7 Convergence of MUCas on Weibo and APS data sets. (A) Weibo (0.5 h), (B) Weibo (1 h), (C) APS (3 years), and (D) APS (5 years) [Color figure can be viewed at wileyonlinelibrary.com]

APS, respectively. We plot the training process of MUCas on the Weibo and APS data set and show the results in Figure 7. Clearly, the training loss of MUCas consistently decreases and converges to a lower value.

(O2) The gap between handcrafted feature-based methods and most deep learning-based baselines are quite small. In some cases the handcrafted feature-based methods even beat some deep learning-based methods. Comparing the Feature-Deep with DeepCas, for example, we can observe that a fully connected layer is enough to achieve competitive results than complicated neural networks (DeepCas) if we have a set of well-designed hand crafted features. However, obtaining such features requiring extensive domain knowledge, which is hard to be generalized to new domains.

(O3) DeepCas—The first deep-learning-based approach for cascade size prediction—performs the worst among the deep learning baselines, because it simply learns the cascade representation based on sampled random walks but ignores temporal and topological information. DeepHawkes, while being successful in modeling temporal information for cascades in a generative learning manner, does not perform well due to its weak ability to learn structural information.

(O4) The rest of the baselines, that is, CasCN, Cascade2vec and VaCas, generate competitive results because they explore structural and temporal information at the same time. When comparing CasCN with Cascade2vec, the performance of Cascade2vec is slightly better, due to the modified convolutional kernel in Cascade2vec, which indeed improves the ability of learning structural features. Besides, VaCas employed VAE[36] to solve the uncertainty problem in structural representation learning and therefore achieves higher accuracy compared with other baselines. Our MUCas not only combines the advantages of CasCN and VaCas, but also takes into account the high-order-scale and position-scale of a cascade graph, leading to a significant improvement in prediction performance.

(O5) When examining the methods with different observation window T, we can observe a general trend, that is, the larger the T, the accurate the predictions. This is intuitive because longer observation time reveals more temporal and structural knowledge regarding information diffusion that helps cascade size prediction.

5.6 | Ablation study (RQ2)

To better investigate the contribution of each scale of information modeled in MUCas, we derive the following variants of MUC as

- -Direction: In "-Direction," we do not consider the directional relation in cascades, that is, regarding the cascade graphs as undirected graphs. We replace the MGN to an vanilla GCN[28] and calculate the normalized adjacency matrix according to $\hat{\mathbf{A}} = \hat{\mathbf{D}}^{-\frac{1}{2}}\bar{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}$.
- -Order: In "-Order," we only focus on 1st-order neighbors in the cascade graphs, that is, setting the max-order number K to 1.
- -Position: In "-Position," we ignore the node's relative position in cascade graph, that is, removing the computation of position information $\mathbf{PW}_p$ in MGN.
- -Dynamic: In "-Dynamic," we remove the sub-graph level influence attention component, and use $\mathbf{G}$ directly.

Figure 8 shows the performance comparisons among MUCas and its variants, which illustrates that: (i) the original MUCas achieves the best performance compared with other variants, demonstrating the motivation of our work, that is, considering the four different scale information for cascade modeling. (ii) From the comparison between "-Direction" and "-Position", we find that effectively modeling the directional relation and node's relative

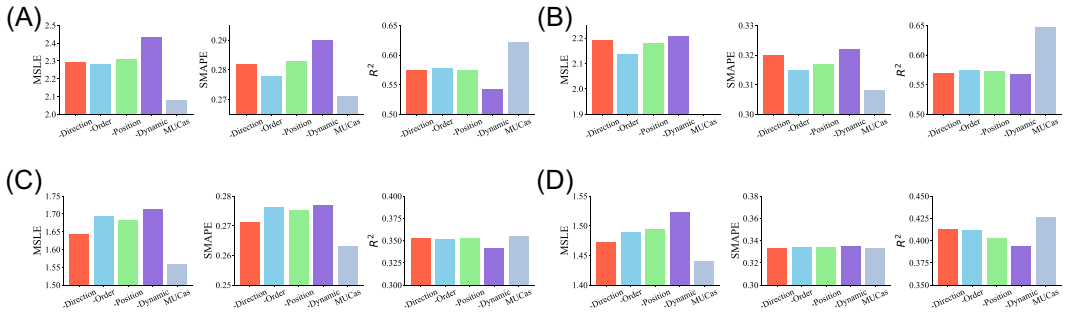


FIGURE 8 Ablation study of MUCas on two data sets. (A) Weibo (0.5 h), (B) Weibo (1 h), (C) APS (3 years), and (D) APS (5 years) [Color figure can be viewed at wileyonlinelibrary.com]

position in the cascade graph will improve the prediction performance. (iii) Removing "-Order" and "-Dynamic" bring a remarkable decrease of the prediction performance, which implies that: (a) nodes with different orders play different importance in prediction task, and (b) the influence decreases as the cascade graph evolves.

To quantify the effectiveness of different levels of capsules, that is, order-level, node-level, and graph-level capsule, we test the model performance by designing several single capsule-based variants of MUCas, including:

Order-level: In "Order-level," we apply sum-pooling to aggregate the order-level capsules $\mathbf{h}_*$ for each node and form node-level capsule $\mathbf{n}_*$. Finally, the sum-pooling operation is employed again to aggregate node-level capsules to form the graph-level capsule $\mathbf{g}_*$.

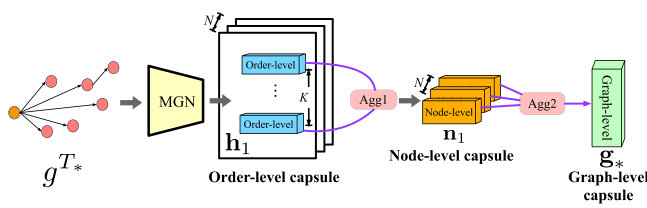
Node-level: In "Node-level," we apply dynamic routing to aggregate order-level capsules $\mathbf{h}_*$ to form node-level capsule $\mathbf{n}_*$. Subsequently, sum-pooling is used to aggregate node-level capsules to form the graph-level features $\mathbf{g}_*$.

Graph-level: In "Graph-level," we apply sum-pooling to generate node-level capsule $\mathbf{n}_*$ from $\mathbf{h}_*$, and then employ dynamic routing to aggregate node-level capsules to form the graph-level capsule $\mathbf{g}_*$.

Figure 9 illustrates the differences between three single capsule-based variants and MUCas. The experimental results are shown in Figure 10, where we can find that: (i) Compared with all single capsule-based variants, the original MUCas performs the best; (ii) Even keeping only one single-level capsule, the model performance is still superior to all the baselines in Tables 2 and 3; and (iii) Compared with original MUCas, the "order-level" variant performs the worst, while the performance of the other two variants (i.e., node-level and graph-level) do not drop a lot. This result demonstrates that (1) the three-level capsules are indispensable, and (2) the dynamic routing is efficient in aggregating features.

5.7 | Hyper-parameter sensitivity (RQ3)

Then we study several important hyper-parameters that may influence the prediction performance of our model. Here we select Weibo data set for experiments and omit APS due to the similar results. The results are shown in Figure 11.



Methods	Agg1	Agg2
MUCas	Dynamic routing	Dynamic routing
Order-level	Sum-pooling	Sum-pooling
Node-level	Dynamic routing	Sum-pooling
Graph-level	Sum-pooling	Dynamic routing

FIGURE 9 Illustration of single capsule-based variants of MUCas [Color figure can be viewed at wileyonlinelibrary.com]

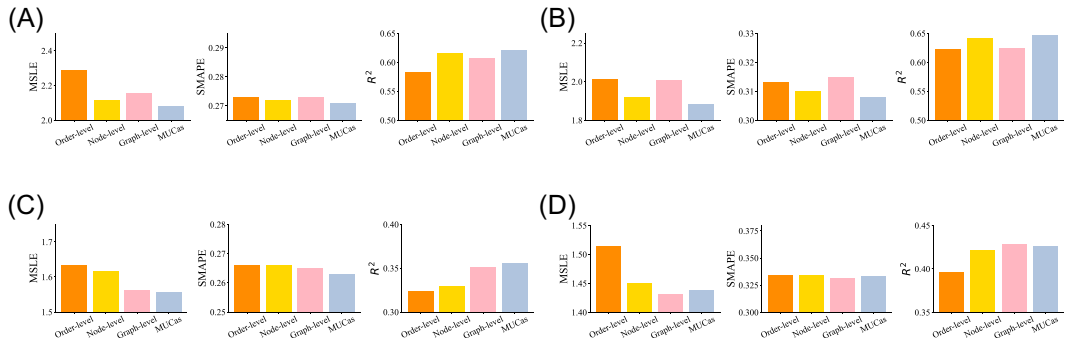


FIGURE 10 Capsule study of MUCas on two data sets. (A) Weibo (0.5 h), (B) Weibo (1 h), (C) APS (3 years), and (D) APS (5 years) [Color figure can be viewed at wileyonlinelibrary.com]

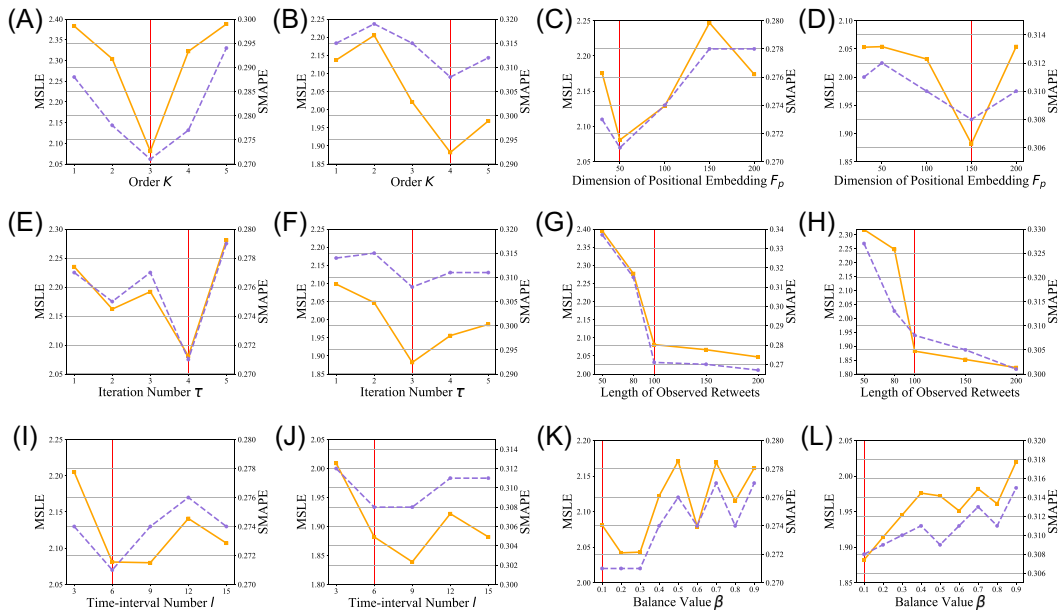


FIGURE 11 Impact of the important hyper-parameters on MUCas. (A) Effect of K ($T = 0.5$), (B) effect of K ($T = 1$), (C) effect of F_p ($T = 0.5$), (D) effect of F_p ($T = 1$), (E) effect of τ ($T = 0.5$), (F) effect of τ ($T = 1$), (G) effect of $|\mathcal{V}^{t_o}|$ ($T = 0.5$), (H) effect of $|\mathcal{V}^{t_o}|$ ($T = 1$), (I) effect of l ($T = 0.5$), (J) effect of l ($T = 1$), (K) effect of β ($T = 0.5$), and (L) effect of β ($T = 1$). Vertical lines are settings we used in previous experiments. Orange solid lines represent the results of MSLE, and the purple dashed lines denote the results of SMAPE [Color figure can be viewed at wileyonlinelibrary.com]

- *Impact of max-order K*: we tried different values of max-order K from 1 to 5, and the prediction results of MUCas are shown in Figure 11A,B. When the observation time is 0.5 h, our MUCas achieves the best performance if $K = 3$. But for 1 h observation, the optimal value of K is 4. The reason is that with the increase of observation time, the information are more likely to propagated to more nodes with deeper depth.
- *Impact of embedding size of positional embedding F_p*: we change the dimensions F_p of positional embedding $\mathbf{P}$ within {30, 50, 100, 150, 200}. Compare the results on 0.5 h with the results on 1 h

(Figure 11C,D), we can see that larger embedding size sometimes may degrade the performance, and the proper embedding size should fall into the scope of [50, 150]. We hypothesize the reason is that most cascade graphs are small, though a few of them may diffuse to a large number of nodes, that is, the 80/20 rule as shown in Figure 4A,B. Therefore, a larger embedding dimensions would incur overfitting issue for those smaller cascade graphs.

- *Impact of iteration number* τ: we tried different values of iteration number τ in dynamic routing, specifically, by increasing the value from 1 to 5. The results are shown in Figure 11E,F, which suggests that increasing the number of iterations would improve the performance first, but deteriorate the model soon. This result raises an open issue in capsule network learning. That is, the appropriate iterations requires careful tuning that makes the capsule network unstable. This issue should be addressed for robust representation learning, which is beyond the scope of this study and left as our future work.
- *Impact of the length of observed retweets* $|\mathcal{V}^{t_o}|$: we track the first 50, 80, 100, 150, and 200 retweets and report the performance of MUCas based on these observed retweets. The experimental results show in Figure 11G,H, where we can observe the improved performance with the increase of observation retweets, which is a natural result of including more training data.
- *Impact of the number of time intervals* l: when sampling sub-cascade graphs from the observed cascade, we set different numbers of time interval, that is, $l = [3, 6, 9, 12, 15]$. The results are shown in Figure 11I,J, which reveal that: (1) the fine-grained sampling does not always perform better. For example, when the observation time is 0.5 h, the performance of MUCas at $l = 6$ is much better than the results when $l = 9, 12$ and 15. This finding also proves our hypothesis in Section 4.1 that the differences between fine-grained sub-graphs become trivial as l grows, which will introduce biases in dynamic modeling. Besides, increasing the number of sub-graphs would significantly increase the computation cost. (2) The choice of the number of time interval heavily depends on the distribution of data set. When the observation window is 0.5 and 1 h, the model performance achieves the best at $l = 6$ and $l = 9$, respectively.
- *Impact of the balance value* β: we change the hyper-parameter β in loss function (Equation 10) from 0.1 to 0.9, and report the results in Figure 11K,L. We can see that a smaller value of β always achieves better performance. Furthermore, $\beta = 0.5$ can be regarded as a watershed, with a value less than 0.5 being far more suitable for model selection than a value greater than 0.5.
- To further support the finding (O5) in Section 5.5, we conducted an extra experiment to assess the model sensitivity varied with the observation time. Specifically, we consider the propagation in the first 15 h because most of the messages in the Weibo data set will decay after this time.[7] The results are shown in Figure 12, where we can clearly see that the longer the observations, the better the performance of the model.

5.8 | Model parameters and computation cost

We compute the time cost of training MUCas and baselines, as well as the required parameters. The results are reported in Table 4. First, the memory required for DeepCas and DeepHawkes is much higher because they need the embeddings of all users in the social network, which required $|U| \times F_u$ parameters—$|U|$ represents the total number of users and F_u denotes the embedding size. Besides, the training time for each epoch of MUCas is around 104*s* and 105*s* in Weibo and APS data set when $T = 0.5$ h and $T = 3$ years, respectively. In contrast, Cascade2Vec is the fastest model that only needs 50s and 70s for Weibo and APS data sets, respectively, although its performance is incomparable.

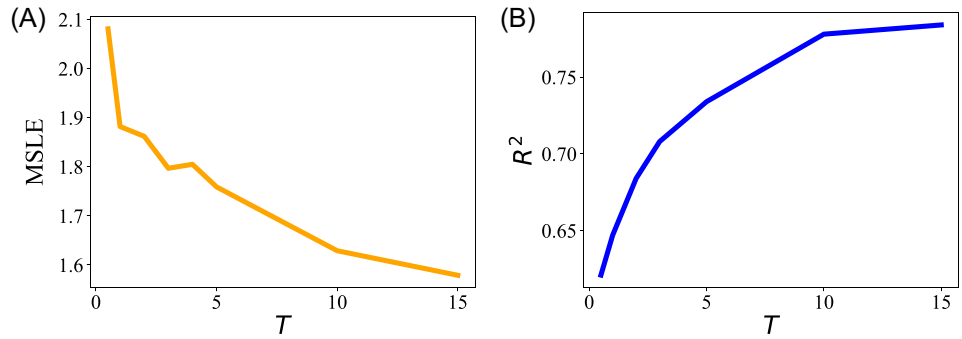


FIGURE 12 The influence of time window in Weibo data set. (A) MSLE and (B) R^2 [Color figure can be viewed at wileyonlinelibrary.com]

TABLE 4 Model parameters and computation time measured by seconds when $T = 0.5$ h and 3 years for Weibo and APS, respectively

Methods	Parameters	Time cost per epoch (s)	
		Weibo ($T = 0.5$ h)	APS ($T = 3$ years)
DeepCas	~250M	~150	~170
DeepHawkes	~250M	~128	~158
CasCN	~278K	~320	~400
Cascade2Vec	~368K	~50	~75
VaCas	~2M	~83	~98
MUCas	~495K	~104	~105

Abbreviation: ~, approximation.

6 | CONCLUSION

In this paper, we proposed a novel cascade prediction model—MUCas, which can capture the multi-scale features regarding information diffusion comprehensively and make good predictions. Specifically, MUCas consists of four components: (1) a time interval-aware sampling layer used to generate sub-cascade graphs from the observed cascade graph, (2) MUG-Caps extracts the direction-scale, position-scale, and high-order-scale information from sub-cascade graphs, (3) attention layer applied to learn dynamic-scale, and (4) a prediction layer to make predictions. We conducted extensive experiments based on two real-world data sets, that is, Weibo and APS. The experimental results demonstrate that our method achieves state-of-the-art performance on information cascade size prediction of tweet propagation in social networks and scientific papers' impact.

As part of our future work, we plan to incorporate hand-crafted features with MUCas, for example, user profiles, content features, etc., which can bring the advantages of feature-based methods into deep learning-based methods. MUCas can be tested on more cascade-related tasks, such as fake news detection,[55] and extend MUCas to other spatial-temporal modeling tasks.[56] Furthermore, modeling the cascades in an unsupervised manner[57,58] can also be considered.

ACKNOWLEDGMENTS

This study was supported by National Natural Science Foundation of China (Grant No. 62072077 and No. 62176043), Sichuan Regional Innovation Cooperation Project (Grant No. 2020YFQ0018), and National Key R&D Program of China (Grant No. 2019YFB1406202).

ORCID

Fan Zhou http://orcid.org/0000-0002-8038-8150

REFERENCES

1. Dow PA, Adamic LA, Friggeri A. The anatomy of large Facebook cascades. ICWSM '13. Seventh International AAAI Conference on Weblogs and Social Media; 2013.
2. AlFalahi K, Atif Y, Abraham A. Models of influence in online social networks. *Int J Intell Syst*. 2014;29(2):161-183.
3. Li C, Ma J, Guo X, Mei Q. DeepCas: An end-to-end predictor of information cascades. WWW '17. Proceedings of the 26th International Conference on World Wide Web; 2017:577-586.
4. Cao Q, Shen H, Cen K, Ouyang W, Cheng X. DeepHawkes: Bridging the gap between prediction and understanding of information cascades. CIKM '17. Proceedings of the 2017 ACM on Conference on Information and Knowledge Management; 2017:1149-1158.
5. Chen X, Zhou F, Zhang K, Trajcevski G, Zhong T, Zhang F. Information diffusion prediction via recurrent cascades convolution. ICDE '19. 2019 IEEE 35th International Conference on Data Engineering; 2019:770-781.
6. Leskovec J, Adamic LA, Huberman BA. The dynamics of viral marketing. *ACM Trans Web (TWEB)*. 2007;1:5-es.
7. Zhou F, Xu X, Trajcevski G, Zhang K. A survey of information cascade analysis: Models, predictions, and recent advances. *ACM Comput Surv (CSUR)*. 2021;54(2):1-36.
8. Cheng J, Adamic L, Dow PA, Kleinberg JM, Leskovec J. Can cascades be predicted? WWW '14. Proceedings of the 23rd International Conference on World Wide Web; 2014:925-936.
9. Jenders M, Kasneci G, Naumann F. Analyzing and predicting viral tweets. WWW '13. Proceedings of the 22nd International Conference on World Wide Web; 2013:657-664.
10. Shen HW, Wang D, Song C, Barabási AL. Modeling and predicting popularity dynamics via reinforced Poisson processes. AAAI '14. Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence; 2014:291-297.
11. Qiu J, Tang J, Ma H, Dong Y, Wang K, Tang J. DeepInf: Social influence prediction with deep learning. KDD '18. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2018.
12. Yang C, Tang J, Sun M, Cui G, Liu Z. Multi-scale information diffusion prediction with reinforced recurrent networks. IJCAI '19. Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence; 2019:4033-4039.
13. Ma J, Gao W, Wong KF. Detect rumors in microblog posts using propagation structure via kernel learning. ACL '17. Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics; 2017:708-717.
14. Ma J, Gao W, Wong KF. Rumor detection on twitter with tree-structured recursive neural networks. ACL '18. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics; 2018:1980-1989.
15. Bian T, Xiao X, Xu T, et al. Rumor detection on social media with bi-directional graph convolutional networks. AAAI '20. Proceedings of the AAAI Conference on Artificial Intelligence; 2020:549-556.
16. Chen X, Zhou F, Zhang F, Bonsangue M. Modeling microscopic and macroscopic information diffusion for rumor detection. *Int J Intell Syst*. 2021;36(10):5449-5471.
17. Wang J, Zheng VW, Liu Z, Chang CC. Topological recurrent neural network for diffusion prediction. ICDM '17. 2017 IEEE International Conference on Data Mining; 2017:475-484.

18. Wang Y, Shen H, Liu S, et al. Cascade dynamics modeling with attention-based recurrent neural network. IJCAI '17. Proceedings of the 26th International Joint Conference on Artificial Intelligence; 2017:2985-2991.
19. Zhou F, Xu X, Zhang K, Trajcevski G, Zhong T. Variational information diffusion for probabilistic cascades prediction. INFOCOM '20. IEEE Conference on Computer Communications; 2020.
20. Tang J, Tang X, Xiao X, Yuan J. Online processing algorithms for influence maximization. SIGMOD '18. Proceedings of the 2018 International Conference on Management of Data; 2018:991-1005.
21. Huang K, Wang S, Bevilacqua G, Xiao X, Lakshmanan LVS. Revisiting the stop-and-stare algorithms for influence maximization. Proc. VLDB Endow. 2017;10(9):913-924.
22. Gou C, Shen H, Du P, Wu D, Liu Y, Cheng X. Learning sequential features for cascade outbreak prediction. *Knowl Inf Syst*. 2018;57(3):721-739.
23. Hochreiter S, Schmidhuber J. Long short-term memory. *Neural Comput*. 1997;9(8):1735-1780.
24. Li C, Guo X, Mei Q. Joint modeling of text and networks for cascade prediction. ICWSM '18. Proceedings of the International AAAI Conference on Web and Social Media; 2018.
25. Liu Z, Wang R, Liu Y. Prediction model for non-topological event propagation in social networks. In: *ICPCSEE '19*. Springer; 2019:241-252.
26. Tang X, Liao D, Huang W, Xu J, Zhu L, Shen M. Fully exploiting cascade graphs for real-time forwarding prediction. AAAI '21. Proceedings of the AAAI Conference on Artificial Intelligence; 2021:582-590.
27. Defferrard M, Bresson X, Vandergheynst P. Convolutional neural networks on graphs with fast localized spectral filtering. NIPS '16. Proceedings of the 30th International Conference on Neural Information Processing Systems; 2016:3844-3852.
28. Kipf TN, Welling M. Semi-Supervised classification with graph convolutional networks. ICLR '17. International Conference on Learning Representations; 2017.
29. Huang Z, Wang Z, Zhang R. Cascade2vec: Learning dynamic cascade representation by recurrent graph neural networks. *IEEE Access*. 2019;7:144800-144812.
30. Xu Z, Qian M, Huang X, Meng J. CasGCN: Predicting future cascade growth based on information diffusion graph. arXiv preprint arXiv:2009.05152; 2020.
31. Wang Y, Wang X, Michalski R, Ran Y, Jia T. CasSeqGCN: Combining network structure and temporal sequence to predict information cascades. arXiv preprint arXiv:2110.06836; 2021.
32. Xu K, Hu W, Leskovec J, Jegelka S. How powerful are graph neural networks? ICLR '19. International Conference on Learning Representations; 2019.
33. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. CVPR '16. Proceedings of the IEEE conference on computer vision and pattern recognition. 2016:770-778.
34. Huang Z, Wang Z, Zhang R, Zhao Y, Zheng F. Learning bi-directional social influence in information cascades using graph sequence attention networks. WWW '20. Companion Proceedings of the Web Conference 2020; 2020:19-21.
35. Cao Q, Shen H, Gao J, Wei B, Cheng X. Popularity prediction on social platforms with coupled graph neural networks. WSDM '20. Proceedings of the 13th International Conference on Web Search and Data Mining; 2020:70-78.
36. Kingma DP, Welling M. Auto-encoding variational Bayes. ICLR '14. International Conference on Learning Representations; 2014.
37. Chen X, Zhang K, Zhou F, Trajcevski G, Zhong T, Zhang F. Information cascades modeling via deep multi-task learning. SIGIR '19. Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval; 2019:885-888.
38. Zhou F, Jing X, Xu X, Zhong T, Trajcevski G, Wu J. Continual Information Cascade Learning. GLOBECOM '20. 2020 IEEE Global Communications Conference; 2020:1-6.
39. Zhou F, Yu L, Xu X, Trajcevski G. Decoupling representation and regressor for long-tailed information cascade prediction. SIGIR '21. Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval; 2021:1875-1879.
40. Abu-El-Haija S, Perozzi B, Kapoor A, et al. MixHop: Higher-order graph convolution architectures via sparsified neighborhood mixing. ICML '19. Proceedings of the 36th International Conference on Machine Learning; 2019:21-29.

41. Xinyi Z, Chen L. Capsule graph neural network. ICLR '19. International Conference on Learning Representations; 2019.
42. Seo Y, Defferrard M, Vandergheynst P, Bresson X. Structured sequence modeling with graph convolutional recurrent networks. ICONIP '18. International Conference on Neural Information Processing; 2018: 362-373.
43. Yao L, Mao C, Luo Y. Graph convolutional networks for text classification. AAAI '19. Proceedings of the AAAI Conference on Artificial Intelligence; 2019:7370-7377.
44. Lei F, Liu X, Jiang J, Liao L, Cai J, Zhao H. Graph convolutional networks with higher-order pooling for semisupervised node classification. *Concurr Comput Pract Exp*. 2020:e5695.
45. Casleton EM, Nordman DJ, Kaiser MS. Local structure graph models with higher-order dependence. *Can J Stat*. 2021;49(2):497-513.
46. Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. NIPS '17. Proceedings of the 31st International Conference on Neural Information Processing Systems; 2017:6000-6010.
47. Liu S, Chen L, Dong H, Wang Z, Wu D, Huang Z. Higher-order weighted graph convolutional networks. arXiv preprint arXiv:1911.04129; 2019.
48. Sabour S, Frosst N, Hinton GE. Dynamic routing between capsules. NIPS '17. Proceedings of the 31st International Conference on Neural Information Processing Systems; 2017:3859-3869.
49. Rosario dVM, Breternitz M Jr, Borin E. Efficiency and scalability of multi-lane capsule networks (MLCN). *J Parallel Distrib Comput*. 2021;155:63-73.
50. Xiao Z, Xu X, Zhang H, Szczerbicki E. A new multi-process collaborative architecture for time series classification. *Knowl-Based Syst*. 2021;220:106934.
51. Liu J, Chen S, Wang L, et al. Multimodal emotion recognition with capsule graph convolutional based representation fusion. ICASSP '20. IEEE; 2021:6339-6343.
52. Peer D, Stabinger S, Rodríguez-Sánchez A. Limitation of capsule networks. *Pattern Recognit Lett*. 2021;144: 68-74.
53. Lin Z, Feng M, Santos CNd, et al. A structured self-attentive sentence embedding. ICLR '17. International Conference on Learning Representations; 2017.
54. Li J, Li S, Zhao WX, et al. Knowledge-enhanced personalized review generation with capsule graph neural network. CIKM '20. Proceedings of the 29th ACM International Conference on Information & Knowledge Management; 2020:735-744.
55. Chen X, Zhou F, Zhang F, Bonsangue M. Modeling microscopic and macroscopic information diffusion for rumor detection. *Int J Intell Syst*. 2021;36(10):5449-5471.
56. Xiao Y, Yin H, Zhang Y, Qi H, Zhang Y, Liu Z. A dual-stage attention-based Conv-LSTM network for spatio-temporal correlation and multivariate time series prediction. *Int J Intell Syst*. 2021.
57. Cheng Z, Wang S, Zhang P, Wang S, Liu X, Zhu E. Improved autoencoder for unsupervised anomaly detection. *Int J Intell Syst*. 2021;36(12):1-23.
58. Zheng W, Yan L, Gou C, et al. Learning to learn by yourself: unsupervised meta-learning with self-knowledge distillation for COVID-19 diagnosis from pneumonia cases. *Int J Intell Syst*. 2021;36(8): 4033-4064.

How to cite this article: Chen X, Zhang F, Zhou F, Bonsangue M. Multi-scale graph capsule with influence attention for information cascades prediction. *Int J Intell Syst*. 2022;37:2584-2611. doi:10.1002/int.22786