IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Cascade2vec: Learning Dynamic Cascade Representation by Recurrent Graph Neural Networks.

ZHENHUA HUANG, ZHENYU WANG, AND RUI ZHANG
School of Software Engineering, South China University of Technology, GuangZhou 510006, China

Corresponding author: Zhenyu Wang (e-mail: wangzy@scut.edu.cn).

This work was supported in part by the Fundamental Research Funds for the Central Universities of China under Grant No. 2017BQ024, in part by the Natural Science Foundation of Guangdong Province under Grant No. 2017A030310428, in part by the Science and Technology Program of Guangzhou under Grant No. 201802010025, in part by the Industry-University Research Fund Project of Guangzhou under Grant No. 2019PT103, and in part by the IBM-CSC Young Data Scientists Program.

ABSTRACT An information dissemination network (i.e., a cascade) with a dynamic graph structure is formed when a novel idea or message spreads from person to person. Predicting the growth of cascades is one of the fundamental problems in social network analysis. Existing deep learning models for cascade prediction are primarily based on recurrent neural networks and representation on random walks or propagation paths. However, these models are not sufficient for learning the deep spatial and temporal features of an entire cascade. Therefore, a new model, called Cascade2vec, is proposed to learn the dynamic graph representation of cascades based on graph recurrent neural networks. To learn more effective graph-level representation of cascades, the current graph neural networks are improved by designing a graph residual block, which shares attention weights between nodes, and by transforming features through perception layers. Furthermore, the proposed graph neural network is integrated into a recurrent neural network to learn the temporal features between graphs. With this method, both the spatial and temporal characteristics of cascades are learned in Cascade2vec. The experimental results show that our method significantly reduces the mean squared logarithmic error and median squared logarithmic error by 16.1% and 12%, respectively, in the cascade prediction at one hour in the Microblog network dataset compared with strong baselines.

INDEX TERMS Social Network, Information Dissemination Network, Cascade Prediction, Graph Neural Networks

I. INTRODUCTION

Online social network platforms facilitate the dissemination of novel ideas, news, and messages [**?**], [**?**], [**?**]. The information trajectories that are transmitted between people form information dissemination trees (i.e., cascades) with different popularity, depth, and shape [**?**], [**?**]. It would be useful to predict the future diffusion popularity or states of a cascade in its early stages. For example, one can have a wiser view in which papers will become viral and recommend them to potential readers.

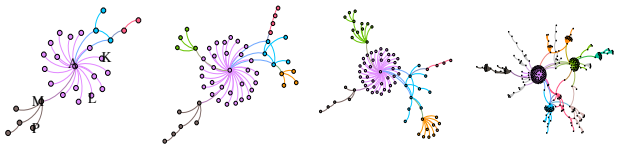
The commonly used abbreviations and notations are listed in the Table **??**. An example of the dynamic structure of a cascade or information dissemination network can be seen in Fig. **??**. Nodes in the same community are presented in the same color by the Louvain community detection algorithm [**?**]. A user in the cascade C_a (see Fig. **??**) posted a microblog at a specific time. Then, other users retweet the microblog and make comments, and edges are formed between the users in the cascade. For example, user M retweeted the microblog, and the microblog was further retweeted by some members (e.g., the node P) of the gray community as shown in Fig. **??**(a). During the message propagation process, the diffusion network structure changes with time as shown in Fig. **??**. Many works have aimed at predicting cascade growth from both theory and practice [**?**], [**?**], [**?**], [**?**], [**?**], [**?**]. There are many challenges in cascade prediction; for instance, some underlying factors related to retweet behaviors are too complicated to be quantified. It is difficult to build powerful models for dynamic graphs. In addition, the popularity cascade roughly follows a power-law distribution,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2942853, IEEE Access

Abbreviations	Description
DeepCas	The cascade prediction method proposed in [**?**].
DeepHawkes	The cascade prediction method proposed in [**?**].
Cascade2vec	The cascade prediction method proposed in our paper.
GNNs	Graph neural networks, refer to [**?**].
GPN	The graph perception network proposed in our paper.
ChebyNet	A graph neural network proposed in [**?**].
GCN	The graph convolutional network proposed in [**?**].
GAT	The graph attention network proposed in [**?**].
DGCNN	A graph neural network proposed in [**?**].
DAGCN	A graph neural network proposed in [**?**].
GIN	The graph isomorphism network proposed in [**?**].
GRUs	The gated recurrent units proposed in [**?**].
C_i	A cascade i.
T	The observation time in cascade prediction.
G_i^t	The graph snapshot of the cascade i at time t.
A	The adjacency matrix of a graph.
X	The feature matrix of a graph.
f	Perception networks.
W	Parameters of neural networks.

TABLE 1: The commonly used abbreviations and notations in this paper.

making the problem even more difficult [**?**].

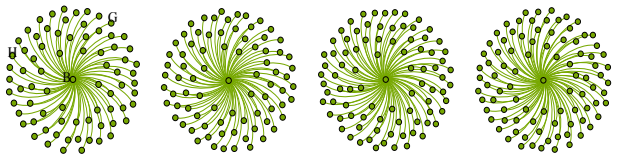


(a) 15 minutes (b) 30 minutes (c) 1 hour (d) 24 hours

FIGURE 1: The cascade C_a with viral structure. (a). 27 retweets. (b). 52 retweets. (c). 83 retweets. (d). 409 retweets.

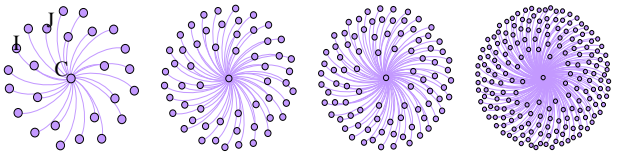
In the works written prior to deep learning or representation learning, effective features including content features, temporal features, and user-related features were mined to train classification or regression models on cascade prediction [**?**], [**?**], [**?**]. Cheng et al. [**?**] proposed the method DeepCas and verified that the end-to-end model is superior to the approaches that use hand-crafted features. The DeepCas applied random walks to sample the spatial information of cascades and used node2vec [**?**] to learn the node representations. But it is not sufficient to learn a good representation only based on spatial information since temporal information also has important impacts on cascade growth. From Fig. **??** and Fig. **??**, we can see that the growth of cascades can be very different even though the two cascades have similar structural shapes. For example, at the time when the messages have been released for an hour, the cascade C_b (see Fig. **??**) and cascade C_c (see Fig. **??**) have equal numbers of retweets.

Cao et al. [**?**] took the temporal features into consideration and designed a new model called DeepHawkes that combines the Hawkes process and deep learning to greatly improve the predictive performance. One of the drawbacks of current deep learning-based cascade prediction models (e.g., DeepCas and DeepHawkes, which are based on random walks or propagation paths) is that they fail to produce an effective graph-level representation of cascades and suffer from learning large-scale parameters. For example, the cascade C_a and cascade C_b have almost equal numbers of retweets at 30 minutes and 1 hour after the message was posted, but the popularity of cascade C_a at 24 hours is 409, far larger than that of cascade C_b, mainly due to their different cascade structural types (i.e., viral structure and broadcast structure, refer to Sharad et al. [**?**]). Therefore, more efficient and effective spatial feature representation methods need to be proposed and applied in cascade prediction. Since at each time point, there is a subgraph as shown in Fig. **??** and Fig. **??**. It motivates us to establish cascade representation models by graph neural networks (GNNs) [**?**], [**?**]. Leveraging on GNNs, the spatial features of graphs are learned as a whole so that we can have a better graph-level representation of a cascade and fewer training parameters in the model.



(a) 15 minutes (b) 30 minutes (c) 1 hour (d) 24 hours

FIGURE 2: The cascade C_b with a broadcast structure. (a). 75 retweets. (b). 79 retweets. (c). 86 retweets. (d). 89 retweets.



(a) 15 minutes (b) 30 minutes (c) 1 hour (d) 24 hours

FIGURE 3: The cascade C_c with a broadcast structure. (a). 35 retweets. (b). 51 retweets. (c). 86 retweets. (d). 184 retweets.

Many graph neural networks have been proposed in recent years. The models include ChebyNet [**?**], GCN [**?**], GAT [**?**], DGCNN [**?**], DAGCN [**?**] and GIN [**?**], etc. However, these networks are still not powerful enough in the graph regression tasks and suffer from over-smoothing and feature loss during propagation [**?**]. So we extend the current GNNs motivated by the idea from ResNet [**?**] and Transformer [**?**], and propose a new graph neural network model, called Graph Perception Network (GPN). The GPN applies a new attention mechanism and residual network to improve the learning ability of graph neural networks, which is conceptually simple but outperforms the previous GNNs in graph classification and graph regression tasks.

To learn the temporal features between subgraphs at different periods, the proposed graph neural network is integrated

into the bidirectional gated recurrent units (GRUs) [**?**] and a graph recurrent neural network is designed. Based on the graph recurrent neural network model, Cascade2vec is introduced to learn both the spatial and temporal features of cascades. Furthermore, we propose a biased mean squared error loss that benefits model training in power-law distributed data. A weighted sum pooling mechanism is also applied to integrate features at different moments.

Instead of learning an embedding feature for each user as in DeepCas [**?**] and DeepHawkes [**?**], Cascade2vec models the cascades as dynamic graphs and incorporates the users' representation features into the representation learned by the graph neural networks. Thus, the number of training parameters can be sharply reduced. Compared to DeepCas and DeepHawkes with tens of millions of parameters, Cascade2vec only has approximately 44, 000 model parameters, which is advantageous in training and applications.

Our contributions in this paper are as follows:

- To improve the graph-level representation of graphs in cascades, we propose a more powerful graph neural network that outperforms current GNNs in graph-level classification and regression tasks. The concept of the residual network and shared attention weights are adopted in the proposed graph neural network GPN to enhance its learning ability of graph representation.
- We model the cascade as a dynamic graph and apply a graph neural network into the cascade feature learning. A dynamic cascade representation model, entitled Cascade2vec, is proposed by incorporating the proposed graph neural network into recurrent neural networks. Compared with DeepHawkes and DeepCas, our model can better represent the spatial and temporal features of cascades. The proposed Cascade2vec achieves significant improvements in cascade prediction over the state-of-the-art methods.

Our codes and processed data will be available at https://github.com/zhenhuascut/Cascade2vec.

II. RELATED WORKS

The related works are discussed from two perspectives, one is cascade prediction and the other is graph neural networks.

Cascade prediction. Existing cascade prediction methods roughly fall into three categories: feature-based approaches, generative approaches, and deep learning approaches.

Feature-based approaches. This kind of methods generally analyze and extract effective features related with popularity growth. Then classifiers or regression models are trained based on the hand-crafted features. The features mainly include temporal features, structural features, content features, and user features. Hong et al. [**?**] used a theme model to learn the topic distributions of text contents in Microblog, and applied it as text features to predict whether a microblog will be popular. Ma et al. [**?**] found that a microblog is more likely to be retweeted by users when its content is related with hot topics. Jenders et al. [**?**] found that content features and user features can be complementary and improve the accuracy of popularity prediction. The initial structural link density is found highly related with cascade popularity [**?**]. The community structure of the relationship network among early adopters is also found to have impacts on the future popularity growth [**?**]. Yi et al. [**?**] found that the entropy of PageRank centrality in early time is closely related to future cascade structure. It is reported that the prediction error can be further reduced when combing temporal features into structure features [**?**]. Cheng et al. [**?**] have pointed out that the cascades can be predicted to certain extents using feature-based approaches. They built cascades from Facebook and verified the effectiveness of temporal features, structural features, content features, and user features. Overall, the performance of feature-based methods highly relies on the extracted features. However, it is costly and inefficient to quantify all the features and design perfect feature patterns in social networks.

Generative approaches. The generative methods are more related to temporal patterns. The models take online popularity growth as a collection of the arrival process of retweets and model the information diffusion process using the theory of Poisson Process [**?**] or Hawkes Process [**?**], [**?**]. Shen et al. [**?**] extended Poisson Process and proposed Reinforced Poisson Process (RPP) to model the three key factors in information diffusion. The Hawkes process is then verified to perform better than the RPP model in popularity growth prediction on the dataset from Sina Microblogs by Bao et al. [**?**]. Zhao et al. [**?**] extended standard Hawkes process and proposed a method called SEISMIC, which models the self-exciting mechanism of each retweet after observing several tweets in the training set as prior knowledge. The SEISMIC uses a power law function to fit the time decay effect in information diffusion. The complicated factors related to cascade growth are far beyond the theory of temporal process, the pure generative approaches cannot achieve well performance in many cases. The recent research tendency of cascade prediction is incorporating the generative approaches into deep learning approaches such as DeepHawkes [**?**].

Deep learning approaches. The deep learning methods are roughly classified into two categories. One category focuses on learning temporal information of cascades. For example, Gou et al. [**?**] proposed a method for predicting cascade popularity based on recurrent neural networks. An attention mechanism on ensemble temporal features is employed in the method. The other category focuses on learning the spatial information. DeepCas [**?**] was proposed to learn spatial features of cascades in representation learning manner. The end-to-end deep learning method was verified to perform better than features-based and generative approaches in cascade prediction. DeepCas uses random walks to sample the spatial information of cascades. The Node2vec [**?**] is applied to learn the node representation. DeepCas does not take full use of the temporal data, while the DeepHawkes [**?**] takes the temporal information into consideration. DeepHawkes significantly improves accuracy by combing the features of temporal and spatial information. DeepCas and DeepHawkes

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2942853, IEEE Access

learn the cascade structure by random walks or propagation paths, not by building models on the entire cascade. They suffer from learning a large number of parameters. We found that users with similar roles share similar characteristics in different cascades. The user features can be integrated into the graph-level presentation of cascades. In this way, the learning parameters can be sharply reduced. Graph neural networks [**?**] are able to establish deep learning model on the graph structured data, suitable for cascade prediction, and have the potential to improve the performance of cascade prediction.

Graph Neural Networks. The graph level representation is one of the basic problems in graph neural network (GNN). The concept GNN was early proposed by Scarselli et al. [**?**]. ChebyNet [**?**] was proposed to classify the graphs based on spectral graph theory [**?**] and Chebyshev polynomials at the conference of NeurIPS 2016. The ChebyNet model shows competitive performance on image and text classification compared with CNN. Its simplified version Graph Convolutional Network (GCN) [**?**] is also a common used graph neural network. The graph attention network (GAT) [**?**] applies concatenation attention mechanism to calculate attention coefficients of neighbors, which improves the learning ability of GCN in node classification tasks. However, we apply a different mechanism that is more suitable in graph-level representation. Zhang et al. [**?**] proposed Deep Graph CNN (DGCNN) that contains a localized graph convolution model with two graph kernels and a SortPooling. To address the problem of current graph neural networks lost information during propagation steps and suffer simple pooling method. Chen et al. [**?**] proposed a dual attention graph convolution network (DAGCN) to learn the node representation and importance by several novel attention mechanism. It is proven that the GNNs are at most as powerful as Weisfeiler-Lehman test in distinguish graph structure [**?**]. A simple neural architecture, Graph Isomorphism Network (GIN), is then proposed based on the injective aggregation and pooling. However, we argue that the current GNNs are still not powerful in graph level representation tasks, (i.e., graph classification and regression tasks). The GNNs are required to be further improved or designed when applied in certain applications, such as graph regression tasks in traffic forecasting [**?**], [**?**]. There is also an urgent need for more effective graph neural networks that are able to produce a better graph-level spatial representation of the information cascades. Current graph neural networks suffer from over-smoothing [**?**]. For example, stacking multiple GCN layers result in over-smoothing, i.e., all nodes will converge to the same semantics. We apply the graph perception network in transforming the features of different layers so that the semantic consistency is maintained and use a residual network to enhance the representation. Shared attention weights and a weighted sum pooling method are also adopted to address the over-smoothing problem.

To learn how the dynamic characteristics of the cascades, a graph neural network is integrated into the recurrent neural networks [**?**]. The proposed Cascade2vec model integrates spatial features and temporal feature learning in the same framework, which is one of the key contributions to cascade prediction.

III. PRELIMINARIES

A. PROBLEM DESCRIPTION

We model a cascade before the observation time as a dynamic graph in the proposed model, Cascade2vec. The snapshot subgraph at each time t is denoted by $G^t = \{V^t, E^t\}$, where the V^t and E^t are sets of nodes and edges, respectively in the graph at time t. For each cascade C_i, a series of subgraphs $\{G_i^t\}, t \in \{0, 1, ..., T\}$ are established, where T is the observation time. The target time (e.g., 24 hours) is marked as F. In Fig. **??**, the T is set to 2 hours, indicating that the observation time is set to 2 hours. The popularities at 1^{st} hour, 2^{nd} hour, F^{th} hour are P_1, P_T and P_F, respectively. The problem of cascade popularity growth prediction is forecasting the future growth of cascade C_i between times T and F using the information in graphs $\{G_i^t\}, t \in \{0, 1, ..., T\}$ before the observation time T. The cascade growth is denoted by $\triangle P$, as shown in Fig. **??**.

B. DATASETS

Microblog Network. The Microblog network dataset is from Cao et al. [**?**]. On the Sina Weibo platform, 119,313 Twitter-like microblogs and their diffusion trees are extracted and built. Examples of the microblog cascades can be seen in Fig. **??** and Fig. **??**. The cascades with less than 10 retweets or more than 1000 retweets during the observation time are filtered, which is the same method as that used by DeepHawkes [**?**]. The observation time T is set to 1 hour, 2 hour and 3 hours, and the target time F is set to 24 hours. When the popularity growth is greater than 10, the distribution of the popularity growth roughly follows the power-law distribution shown in Fig. **??** (1). In the case of a popularity growth is less than 10, the number of cascades is almost equal, a little different from the popularity distribution of total cascades.

APS Citation Network. The APS citation network dataset includes the citation relationships between papers from 1893 and 2017 and is provided by American Physical Society (APS). The papers from 1893 to 1997 are selected to build cascades so that each of the papers is allowed to develop for at least 20 years. The distribution of popularity growth of the citation network shown in Fig. **??** (2) is quite similar to that of Microblog network. The structure of a citation cascade is shown in Fig. **??** (a), which is quite similar to the cascades in the Microblog network. We set the observation time to 5 years, 7 years and 9 years. The target time is set to 20 years.

The total number of users and edges in the Microblog network when T is set to 1 hour and the APS citation network when T is set to 5 years are listed in Table **??**. Note that users can be duplicated when we build cascades since they may be involved in forwarding different messages.

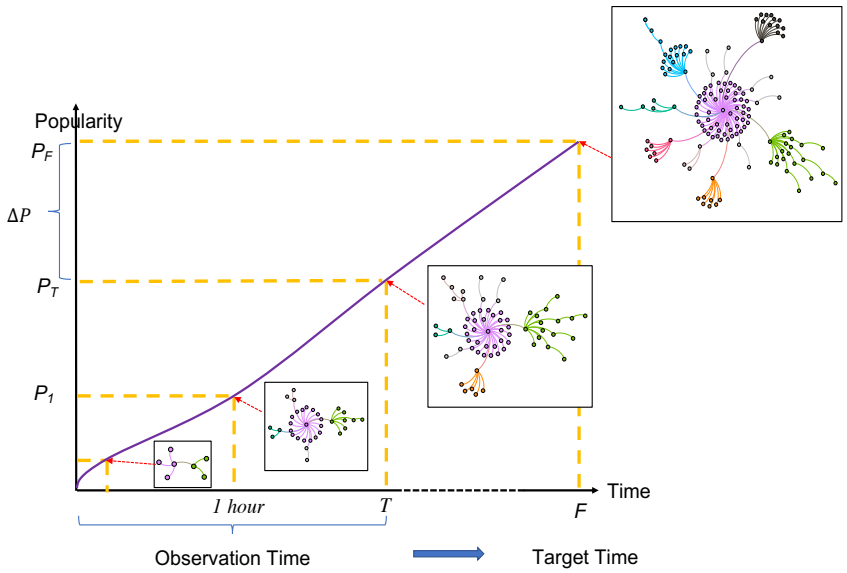
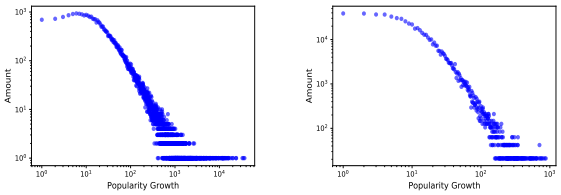
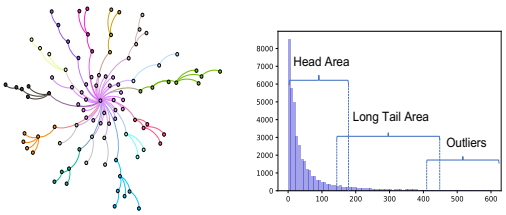


FIGURE 4: The problem of cascade popularity growth prediction.



(a) Microblog (b) APS Citation

FIGURE 5: Distributions of popularity growth.



(a) (b)

FIGURE 6: (a) An example of the citation cascade. (b) The long tail distribution of cascade popularity in the Microblog network.

C. EVALUATION

The distribution of target values roughly follows a power-law distribution; i.e., there is a long-tail area in the data distribution, as shown in Fig. **??** (target values greater than 600 are not represented to produce a better visualization). It shows that cascades with larger growth tend to have fewer popularities while contributing more on loss. We use $log2(y+1)$ as the target instead of the direct value, where y is the count of retweet growth or citation growth. The mean squared logarithmic error (MSLE) is adopted as the loss and function for evaluation. The MSLE is defined as $MSLE = \frac{1}{n}\sum_i^n SLE^i$. The loss is sensitive with the distribution. Therefore, we modified the loss with a bias in the following: $bSLE = (y/y_{max} + 1.0) * (y - \hat{y})^2$ where $\hat{y}$ is the predicted value and y_{max} is the maximum value of the targets. The standard squared logarithmic error (SLE) is replaced by the bSLE in the training steps. In the evaluation steps, the SLE is retained. When y is small and falls into the head area with rich data as shown in Fig. **??**, the deviation is close to the value of a standard MSE. When the y is in the long-tail area, it receives more attention than the data in the head area. We also use the median SLE (mSLE) as one of the evaluation metrics since the MLSE is sensitive to long-tail area and outliers. The definitions of the Head Area, Long-Tail Area, and Outliers are not a key point of our paper and are left to future works. More discussions about the power-law distribution can be seen in the book [**?**]. To evaluate the performance of GNNs in regression tasks, the R2score (i.e., the coefficient of determination) is also applied. However, the R2score is sensitive to noisy data, so it is not used in all the experiments.

TABLE 2: Basic information of the Microblog network and the APS citation network.

	Microblog Network	APS Citation Network
# Nodes	3.45 million	1.20 million
# Edges	6.33 million	3.34 million
# Selected Cascades	40, 826	23, 356
# Edges / Nodes	1.82	2.78

D. CASCADE PREDICTION BASELINES

Features-Linear (F-Linear). Temporal features, structural features, and features of early adopters introduced by Cao

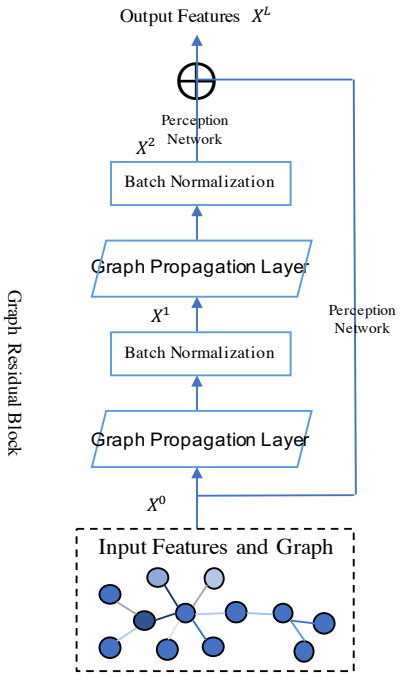


FIGURE 7: The Structure of Graph Residual Block.

et al. [**?**] are extracted in the **F-Linear** method. After the features have been calculated, a multi-perception layer is applied to train a linear regression model.

DeepCas [**?**] learns the spatial features of cascade graphs in an end-to-end manner. It mainly utilizes the GRUs and an attention mechanism to extract the structural information and node identities in cascade prediction. The method targets on learning spatial features while ignoring the temporal features.

DeepHawkes [**?**] is a state-of-the-art cascade prediction method. It builds a bridge between the Hawkes process and recurrent neural networks. The method learns the factors of the Hawkes process in a data-driven manner and outperforms DeepCas [**?**] and SEISMIC [**?**].

IV. CASCADE2VEC

A. GRAPH PERCEPTION NETWORK

As mentioned above, the GPN inherits GIN [**?**], GCN [**?**], and ResNet [**?**] while having a more powerful learning ability of graph-level representations. The basic component of the GPN is shown in Fig. **??**, which is called as *Graph Residual Block* since the idea is motivated by ResNet [**?**]. The input feature $X^0 \in \mathbb{R}^{N*F}$ is converted into X^L after two graph propagation layers. The graph propagation layer performs graph convolutional operations and features aggregations similar to those in GCN [**?**]. The output of the last graph propagation layer and the input features are converted by perception layers and then concatenated to produce the final output features. The output features are further connected by perception layers to perform classification or regression tasks in certain problems.

The process of the *Graph Residual Block* is described in the following:

Input: $X^0 \in \mathbb{R}^{N*F_0}, A \in \mathbb{R}^{N*N}$.

Output: X^L.

where N is the number of graph nodes, F_0 is the dimension of the input features, and $L = 2$ is the number of graph propagation layers. In the *graph propagation layer*, the features of node v in layer l propagate among neighbors and are updated by Equation **??**:

$$h_v^l = ReLU(\sum_{u \in N(v) \bigvee \{v\}} f_{MLP}^l(h_u^{l-1}) * \alpha_{uv}) \quad (1)$$

$$\alpha_{uv} = Softmax_u(e_{uv}) = \frac{exp(e_{uv})}{\sum_{k \in N(v) \bigvee \{v\}} exp(e_{kv})} \quad (2)$$

$$e_{uv} = ReLU((f_{MLP}^l(h_v^{l-1}))^T W_a^l f_{MLP}^l(h_u^{l-1})) \quad (3)$$

where $N(v)$ are the neighbors of node v, $W_a^l \in \mathbb{R}^{F^l * F^l}$ represents shared attention weights in the layer, and F^l is the output dimension of f_{MLP}^l in the l^th propagation layer. The α_{uv} indicates the strength of the features that should be adopted by the neighbors. The f_{MLP}^l is a two-layer perception network in the layer l. According to Xu et al. [**?**], the sum aggregator is more likely to produce an injective representation. However, it still leads to over-smoothing. Different neighbors have different impact strengths, so shared attention weights are adopted in the GPN. The graph propagation layer is followed by a batch normalization layer [**?**].

The output feature vector for node v by the graph residual block is: $h_v^{L'} = ReLU(f_P^0(h_v^0) + f_P^L(h_v^L))$

where f_P^0 and f_P^L are one-layer perception networks that convert the features into the same dimensions so that they can be concatenated as residual networks [**?**]. To obtain representation features h_G for the entire graph, a sum pooling (SumPool) is applied and followed by a perception network f_P^G that converts the presentation of a graph into the targeted dimension. h_G is calculated by Equation **??**. The sum pooling can also be replaced by the SortPooling, which was proposed by Zhang et al. [**?**]. Then, we will verify the performance of the GPN in the graph classification tasks described by Pinar et al. [**?**] and the graph regression tasks introduced in our paper.

$$h_G = f_P^G(SumPool(\{h_v^{L'} | v \in V)\}) \quad (4)$$

B. GRAPH FEATURES

When the nodes have tagged input features, the input feature matrix X is directly used in the model. When there are no input features for the nodes, we extract node features by a graph Fourier transform or degree matrix of the input graphs. The adjacency matrix of a graph is denoted by A. W is the weighted adjacency matrix, and D is the diagonal degree matrix. The graph Laplacian matrix of an undirected graph is calculated by $L = D - W$. The normalized Laplacian matrix is $L = I - D^{-1/2}WD{-}1/2$. If the graph is directed, the normalized directed Laplacian matrix is calculated by

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2942853, IEEE Access

$L = I - (\Phi^{1/2} P \Phi^{-1/2} + \Phi^{-1/2} P^T \Phi^{1/2})/2$, where P is the transition probability matrix and Φ is a matrix with the Perron vector of P on the diagonal [**?**]. L is a positive semi-definite $N \times N$ matrix. We can calculate N eigenvalues $\Lambda = [\lambda^0, ..., \lambda^{N-1}]$ and N corresponding eigenvectors $U = [u^0 ..., u^{N-1}]$. The eigenvectors are real and orthonormal. They are regarded as graph Fourier basis functions [**?**]. The Laplacian matrix L is thus decomposed by $L = U^T \Lambda U$. The features X_t of input graph G^t at time t are calculated by $U^T x$. In this paper, the graph features with top-k (set to 100) eigenvalues are selected.

C. GRAPH-LEVEL REPRESENTATION

In the following experiments, we will compare GPN with current GNNs to verify its effectiveness in spatial representation learning. The output size of f^l_{MLP} in the layers is set to the same value among {16, 32 }. The learning rate is chosen from {5e-4, 1e-4, 1e-3, 1e-2}. The output dimension of the graph residual block is set to 32. The output dimension of F^l is kept to 32. The hidden size and output dimension of multi-layer perception in the GPN is kept the same. The optimizer RMSprop [**?**] is applied. However, in the following experiments, the GPN model is not very sensitive to these parameters in most cases. We only use one graph residual block since it is powerful enough in the experiments. For the comparison methods GCN and GAT, we use two GCN or GAT convolutional layers, respectively, and the hidden size and output dimension are chosen from {16, 32}. For GIN, DGCNN and DAGCN, we used the parameters mentioned in the original papers [**?**], [**?**], [**?**].

1) Graph Classification Task.

Given a set of graphs $\{G_i\}, i \in \{1, ..., n\}$ and their labels $\{Y_i\}, i \in \{1, ..., n\}$, the graph classification task aims to predict the correct class label Y_i of a graph G_i using the learned graph representation h_{G_i}. The datasets of MUTAG, PROTEINS, PTC, COLLAB, and IMDB-M [**?**] are benchmark datasets used in graph classification [**?**]. The MUTAG, PROTEINS, and PTC datasets are bioinformatics data with discrete labels. The COLLAB dataset is a scientific collaboration dataset, and the IMDB-M dataset is a movie collaboration dataset. The average precision and standard deviation from the 10-folder cross-validation are reported in Table **??**. From the table, it is obvious that GPN outperforms baselines. The GAT is better than the GCN and is a very competitive approach. DGCNN and DAGCN are also competitive methods, but they do not perform well in the COLLAB and IMDB-M datasets in which the nodes do not have tag features. The results indicate that the proposed GPN has a stable and powerful learning ability in graph-level representation.

2) Simple Graph Regression Task.

A simple graph regression task is to predict the number of nodes of each graph without any other additional data. The Microblog network dataset is used, and the target is number of nodes of the cascades, which is different from cascade growth prediction. The deep learning models suffer from very large variance if the target value follows a power-law distribution. The target value y (i.e., the number of nodes in a graph) is transformed by $log_2(1 + y)$. As shown in Table **??**, the GPN model has a relatively higher performance than strong baseline models such as GCN, and GIN on the graph regression task of identifying how many nodes are in a graph. The performances of GPN, DGCNN and DAGCN are quite close in terms of R2score. When the true-versus-predictions figures are drawn as shown in Fig. **??**, it can be seen that the GPN achieves an almost perfect performance in the perception of how many nodes in a graph. The strong baseline GIN cannot adopt to certain graphs, especially when the targets become bigger.

TABLE 3: Performance of the GNNs on the graph classification tasks.

	MUTAG	PROTEINS	PTC	COLLAB	IMDB-M
GCN	85.6±5.8	76.0±3.2	64.2±4.3	79.0±1.8	51.9±3.8
GAT	89.5±4.5	78.1±2.8	65.1±5.2	80.6±1.2	53.6±3.7
GIN	89.0±6.0	75.9±3.8	63.7±8.2	80.1±1.9	52.1±3.6
DGCNN	85.8±1.7	75.5±0.9	58.6±2.4	73.7±0.5	47.8±0.8
DAGCN	87.2±6.1	76.3±4.3	62.9±9.6	67.6±2.3	43.6±2.7
GPN	**92.6±3.9**	**78.8±2.7**	**66.8±4.2**	**82.2±1.1**	**54.0±2.5**

TABLE 4: Experiments for predicting how many nodes are in a graph.

	MSLE	R2Score
GCN	0.191	0.896
GIN	0.155	0.933
GAT	0.116	0.946
DGCNN	0.026	0.978
DAGCN	0.028	0.986
GPN	**0.0045**	**0.998**

3) Cascade Growth Prediction.

Then, we introduce a more complicated task related to cascade growth prediction and verify whether the proposed graph neural network is able to learn effective spatial representation features of a cascade, just like Word2vec [**?**] in natural language processing problems. We use G^T_i to represent the subgraph of cascade C_i at time $T = 1$ hour in the Microblog network dataset. Several approaches are applied to learn the representation features of G^T_i. Then, the features are used to predict future growth at the 24th

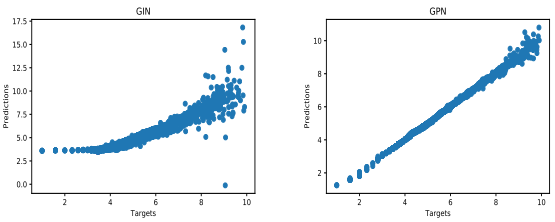


FIGURE 8: Prediction verse targets for the simple graph regression task.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2942853, IEEE Access

TABLE 5: Cascade growth prediction only based on embedding features of the graph at the predicting time.

	MSLE	mSLE
F-Linear	3.70	1.06
GCN	3.61	0.92
DeepCas	3.63	0.81
GAT	3.49	0.85
GIN	3.45	0.88
DGCNN	3.38	0.78
DAGCN	3.42	0.83
GPN	**3.18**	**0.76**

hour of the cascades. Baseline methods include the F-Linear, DeepCas [**?**], GIN [**?**], GCN [**?**], and the proposed GPN. The prediction results are recorded in Table **??**. The results show that the proposed method GPN achieves a 12.4% error reduction in terms of the MSLE compared to the strong baseline DeepCas, while the GCN performs similarly to DeepCas. DGCNN and DAGCN perform better than GCN and GAT on the MSLE. It is noted that the mSLE does not decrease monotonically with the MSLE. The above findings indicate that the proposed graph neural network is capable of producing a better representation of the cascade structure and is conducive to future cascade popularity growth.

4) Viral Structure Learning

As mentioned in Section **??**, the viral structure has significant impacts on cascade popularity growth. The viral structure of cascades has been extensively discussed in [**?**], [**?**], [**?**], [**?**]. From previous works, the cascades with the viral structure have a Winer index higher than 2.5 or a Modularity greater than 0.3 [**?**], [**?**]. We use the Wiener index to calculate the structural virality of cascades. We would like to verify whether the existing models are able to extract effective features of cascades with viral structure. There are 4490 cascades with viral structure (i.e., a Wiener index greater than 2.5), accounting for 11.7% of the Microblog network dataset. In total, 90% of the viral structural cascades are selected as the training set, and 10% of them are selected as the testing set. The prediction on these cascades is more challenging. The average and median popularity growth in the cascades with viral structure are 80 and 245, respectively. While the average and median values of all the cascades are 26 and 114, respectively. The finding indicates that the viral structure facilitates future popularity growth of cascades.

The results in Table **??** show that our method performs even better in cascades with viral structure in terms of the MSLE while achieving a slightly lower performance on the mLSE. The path-sampling method DeepCas produces much worse results on cascades of viral structure. The F-Linear method produces better results than DeepCas, which may because the viral structure actually provides more information about how the cascading structure is formed and changes compared to the broadcast structure [**?**]. Based on the viral structure, it is possible for the models to establish or learn more effective features about the cascades. Overall, the above experiments show that the proposed graph neural network

TABLE 6: Cascade growth prediction among cascades with viral structure.

	MSLE	mSLE
F-Linear	3.52	1.15
DeepCas	3.72	1.32
GPN	**2.83**	**0.83**
Reduction	23.9 %	37.1%

GPN has an effective learning ability for spatial features and graph-level representation .

D. DYNAMIC CASCADE REPRESENTATION

In the following sections, we will discuss how Cascade2vec incorporates the graph perception network into recurrent neural networks to build a dynamic cascade representation model. The whole framework of the dynamic cascade representation model is shown in Fig. **??**. The feature matrix and adjacency matrix of the subgraph at time t_i are denoted by X_i and A_i, respectively. From Fig. **??**, the structure of a cascade changes with time. For example, three red nodes joined the cascade at time t_2, and three green nodes are involved in the cascade at time t_3. Three gates are applied to each moment in our model.

First, the reset gate r_t is calculated by

$$\begin{aligned} r_t &= \sigma(W_{ir} * GPN(X_t, A_t) + b_{ir} + \\ &\quad W_{hr} * h_{t-1} + b_{hr}) \end{aligned} \tag{5}$$

where σ is the sigmoid activation function. $W_{ir} \in \mathbb{R}^{H*K}$, $W_{hr} \in \mathbb{R}^{H*H}$ and $b_{ir}, b_{hr} \in \mathbb{R}^{H}$, K is the output dimension of GPN, and H is the hidden size of the recurrent process in Cascade2vec.

The update gate z_t is computed by

$$\begin{aligned} z_t &= \sigma(W_{iz} * GPN(X_t, A_t) + b_{iz} + \\ &\quad W_{hz} * h_{(t-1)} + b_{hz}) \end{aligned} \tag{6}$$

The hidden state of h_t is then updated by the following equations:

$$\begin{aligned} \tilde{h_t} &= \tanh(W_{in} * GPN(X_t, A_t) + b_{in} + \\ &\quad r_t * (W_{hn} * (h_{(t-1)}, A_t) + b_{hn})) \end{aligned} \tag{7}$$

$$h_t = (1 - z_t) \odot \tilde{h_t} + z_t \odot h_{(t-1)} \tag{8}$$

where $\odot$ is the element-wise product. W_{in}, W_{iz} and W_{iz} have the same shape. W_{hn}, W_{hz} and W_{hr} have the same shape. All the biases have the same shape. The recurrent neural network outputs a hidden representation h_t of the cascade at moment t. Note that the f^l_{MLP}, W_a, W_i and W_h are independent of the number of nodes, so the number of parameters is sharply reduced compared with DeepCas and DeepHawkes. The proposed Cascade2vec is also flexible and easy to be integrated with any kind of recurrent neural networks (e.g., LSTM [**?**] and Transformer [**?**]).

We can simply use the last output h_T as the embedding features of a cascade. The learned h_T contains the temporal information of each moment. However, the outputs at different times may also contain some temporal information. We

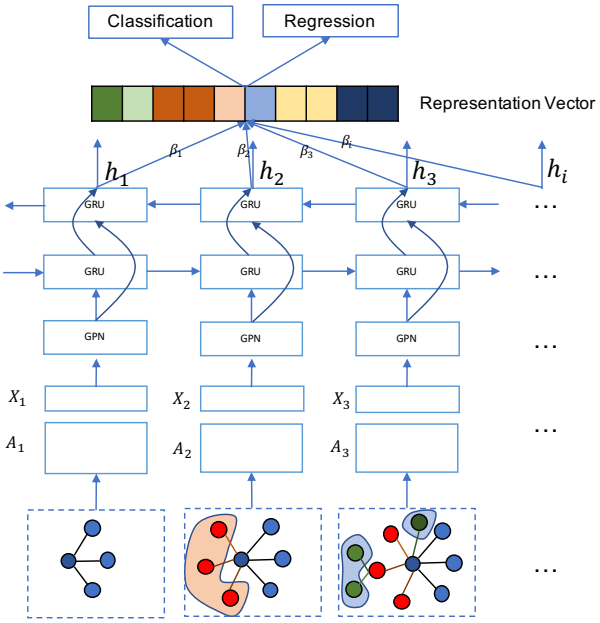


FIGURE 9: The framework of Cascade2vec.

apply a weighted sum pooling method to integrate features at each time. The weighted sum pooling learns a parameter β_t for each moment, and the final representation h_{c_i} of a cascade C_i is computed by:

$$h_{c_i} = \sum_{t=0}^{T} \beta_t * h_t \tag{9}$$

Then, h_{c_i} is connected with a two-layer perception network MLP_{output}. The model outputs a predicted value $\hat{y}$ of cascade C_i using the following equation. Then the MLSE loss will be calculated. Note that y is logarithmic value.

$$\hat{y_i} = MLP_{output} h_{c_i} \tag{10}$$

$$loss(\hat{y_i}, y) = (\hat{y_i} - y)^2 \tag{11}$$

There are several more super-parameters in the Cascade2vec. The hidden size H of GRU is set to 32. The observation time is split every 180 seconds in the Microblog network and every 91 days in the APS citation network.

V. PERFORMANCE OF CASCADE2VEC

The performance of Cascade2vec is verified on the Microblog network and the APS citation network. The relative reduction (denoted by Reduction) is calculated by $|E_b - E_c|/E_b$, where E_b is the samllest error of the compared methods, and E_c is the error from Cascade2vec.

A. PREDICTION PERFORMANCE

According to the results in Table **??** and Table **??**, it can be clearly seen that Cascade2vec significantly improves the performance and reduces the prediction errors. Compared to DeepHawkes, the MSLE is reduced by 16.1% in one hour in the Microblog network dataset. As one of the advanced deep learning methods for cascade prediction, DeepHawkes achieves a quite good performance. However, since DeepHawkes does not model the entire structure of graphs, its prediction performance can be further improved. When the observation time is increased from 1 hour to 2 hours and 3 hours, the relative reduction is decreased. As more information becomes available, the prediction tasks become easier, so the performance differences between Cascade2vec and the baselines become smaller. There is a large margin between the strong baselines and Cascade2vec in the mSLE, indicating that the proposed method is able to learn more useful features that are conducive to the growth of most cascades. Compared with results on the Microblog network, Cascade2vec achieves a relatively smaller improvement on the MSLE in the APS citation network. While in terms of

TABLE 7: Cascade growth prediction results on Sina Weibo.

	MSLE			mSLE		
T	1hour	2hour	3hour	1hour	2hour	3hour
F-Linear	3.701	3.365	3.328	1.058	1.105	1.139
DeepCas	3.631	3.213	3.105	0.808	0.837	0.902
DeepHawkes	2.448	2.279	2.223	0.650	0.675	0.639
Cascade2vec	**2.055**	**1.992**	**1.951**	**0.572**	**0.554**	**0.536**
Reduction	16.1%	12.6%	12.2%	12.0%	17.9%	16.1%

TABLE 8: Cascade growth prediction results on APS citation network.

	MSLE			mSLE		
T	5year	7year	9year	5year	7year	9year
F-Linear	1.582	1.508	1.456	0.679	0.674	0.722
DeepCas	1.629	1.538	1.462	0.671	0.603	0.662
DeepHawkes	1.510	1.337	1.211	0.652	0.584	0.605
Cascade2vec	**1.409**	**1.310**	**1.192**	**0.588**	**0.555**	**0.561**
Reduction	6.7%	2.0 %	1.6 %	9.7%	4.9 %	7.2%

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2942853, IEEE Access

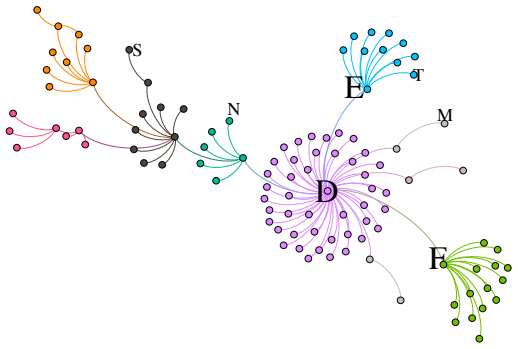


FIGURE 10: An example cascade C_d with viral structure.

the median squared log-error, Cascade2vec still performs much better than the baselines. Overall, the results verify that the graph recurrent neural network is suitable and stable for popularity prediction.

B. VIRAL CASCADE PREDICTION

We will discuss the performance of Cascade2vec on the viral cascade to show its ability in dynamic spatial feature learning and discover reasons why Cascade2vec outperforms baselines. An example cascade with a viral structure in the Microblog network is represented in Fig. **??**. The tweets of the viral cascades disseminate among users, similar to a virus propagating among people. Many branches are formed during the dissemination process, indicating that the tweets have been spread through different communities. We have demonstrated in Section **??** that the GPN is superior to DeepCas on spatial feature learning of cascades. We will evaluate the performance of DeepHawkes and Cascade2vec on cascades with viral structure. The cascades with viral structure are selected in the same way introduced in Section **??**. As shown in Table **??**, Cascade2vec outperforms DeepHawkes in both the MSLE and mSLE. The relative error reduction is even increased to 23.2% on the MSLE and 33.2% on the mSLE. It shows that cascades with viral structure provide more spatial information. However, DeepCas and DeepHawkes fail to make full use of spatial information in cascades. The DeepHawkes is more stable in this case, while the median error is much lower than the error of the total dataset. Although the prediction task in the viral cascades is more challenging, as described above, the performance of our approach is still stable in this part of the dataset. Another reason why DeepHawkes and DeepCas do not perform well on cascades with viral structure is the lack of sufficient datasets. They train an embedding vector for each user. However, we found that the roles users play in information propagation are quite similar among different cascades. For example, the nodes A and M in the cascade C_a and the nodes D, E, and F in the cascade C_d are opinion leaders in their communities.

TABLE 9: Cascade prediction on the cascades with viral structure between DeepHawkes and Cascade2vec .

	MSLE	mSLE
SEISMIC	5.864	1.537
DeepHawkes	2.530	0.845
Cascade2vec	**1.942**	**0.564**
Reduction	23.2%	33.2 %

Many marginal nodes represent ordinary users, such as the nodes N, M, S, and T in the cascade C_d, the nodes P, K, and L in the cascade C_a, the nodes G and H in the cascade C_b, and the nodes I and J in the cascade C_c. We do not learn a specific embedding for each node but rather learn a whole representation of the graphs. Thus, the number of parameters in our model is much less than that in DeepCas and DeepHawkes, while the performance is more stable when the data is not sufficient.

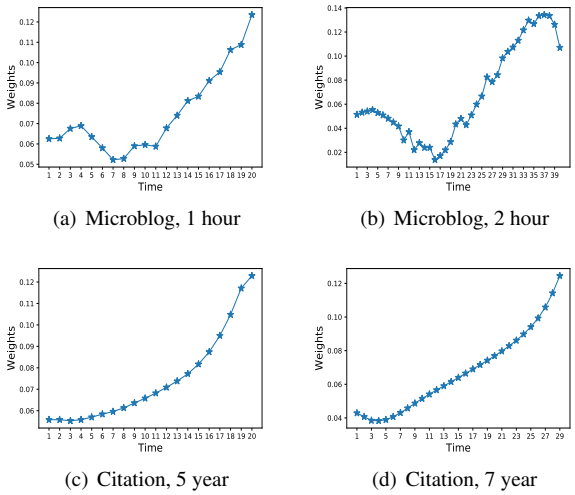
Although the viral structure provides more spatial information about cascades, temporal characteristics still play a significant role in cascade prediction. In the Microblog network dataset, the MSLE of the GPN that is only based on the static structural information of cascades at 1 hour is 3.18, while the MSLE of Cascade2vec is 2.055, which demonstrates the importance of the temporal features of the graphs. However, if only the temporal information is taken into consideration in a model, it also cannot achieve a high performance either. For instance, SEISMIC [**?**], an implementation of a Hawkes self-exciting point process, is a temporal model that applies the self-exciting mechanism for each retweet. It also uses a power-law function to fit the time decay effects in information diffusion. The SEISMIC suffers from noisy data and performs worse in cascade prediction, as shown in the Table **??**. Therefore, a better way for cascade prediction is integrating both the spatial and temporal features.

C. TEMPORAL DYNAMICS

We present how the temporal weights change with time in Fig. **??**. In the Microblog network data, we split the time of 1 hour into 180-seconds sections. There is a temporal weight parameter at each moment. From the weights, we can see how different moments contribute to the final popularity growth. At one hour in the Microblog network dataset, the 20 parameters are shown in Fig. **??** (a). The weights decrease slightly at the first several splits and then increase until the observation time of 1 hour. This finding shows that the features of several early splits and recent features have more important impacts on the future popularity growth of microblogs. When the observation time is set to 2 hours, similar effects occurred during cascade prediction. In the APS citation network dataset, features that are close to the observation time are given higher weights. However, there is a difference in the temporal dynamics between the Microblog network and the APS citation network. The time weight in the APS citation network increases more smoothly than that in the Microblogs network. The influences of retweet behavior and popularity growth in microblogs are more complicated.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2942853, IEEE Access

The mood of users, the content of the microblogs, and the publishing time can significantly impact the popularity growth of tweets. While the citation growth in most cases is a slow process, the trend in the temporal weights is more smooth.



(a) Microblog, 1 hour (b) Microblog, 2 hour

(c) Citation, 5 year (d) Citation, 7 year

FIGURE 11: Temporal weights in cascade prediction.

TABLE 10: Model parameters and outputs of Cascade2vec.

Layers	Model Parameters	Output Size
Input Layer	0	N*100
Perception Layer f_{MLP}^{l}	100*32+32*32	N*32+N*32
Activation	0	N*32
Attention Layer	32*32	N*32
Activation	0	N*32
Batch Normalization	32	N*32
Activation	0	N*32
Perception Layer f_{MLP}^{l}	32*32+32*32	N*32+N*32
Activation	0	N*32
Attention Layer	32*32	N*32
Activation	0	N*32
Batch Normalization	32	N*32
Activation	0	N*32
Perception Layer f_{P}^{0}	100*32	N*32
Perception Layer f_{P}^{L}	32*32	N*32
Residual Layer	0	N*32
Activation	0	N*32
Perception Layer	32*32+32*32	N*32+N*32
Activation	0	N*32
GRU-Wi	32*32*3*2	N*32*3*2
GRU-Wh	32*32*3*2	N*32*3*2
GRU-Reverse-W_i	32*32*3*2	N*32*3*2
GRU-Reverse-W_h	32*32*3*2	N*32*3*2
temporal weights	20	N*64
Perception Layer	64*64	N*64
Activation	0	N*64
Output Layer	64	N*1
Total	43, 412	N*1, 733
Memory Requirements	340 KB	13.5 MB

D. COMPLEXITY ANALYSIS

We analyze the computational complexity refer to the DAGCN [**?**]. The computational complexity of the GPN is related to the density of input graphs. Each node has to perform feature propagations with its neighbors. Suppose the average number of neighboring nodes is c. Then, there are $2c * N$ propagations in total, where N is the number of nodes, i.e., $|V|$. In the worst case when the input graph is very dense, c and N are on the same order of magnitude. When the graphs are sparse, c is a small constant. Therefore, the time complexity of GPN is $O(N^2)$ in the worst-case scenario. Since the graphs of cascades are sparse, the time complexity of the GPN is $O(N)$ in most case. From Table **??**, we can see that the value of c is very small in cascades. For Cascade2vec, it applies GPN at each time segment. Suppose there are S temporal segments, where S is a small constant integer; e.g., $S = 20$ when $T = 1$ hour in the Microblog network dataset. Therefore, the time complexity of Cascade2vec is $O(2c * N * S)$=$O(N)$ in cascade prediction [**?**].

Space complexity analyses of deep learning models can be complicated. Rarely papers perform a space complexity analysis of graph neural networks or cascade prediction models [**?**], [**?**], [**?**], [**?**]. Once a deep learning model has been established, the parameters are fixed. The parameters and the required memory of the GPN and GRUs do not grow with the inputs in Cascade2vec. The space complexity of our model should be $O(1)$ following the analysis in [**?**], [**?**]. However, it does not make sense to analyze the space complexity of the model parameters. Therefore, we also analyze the outputs in each layer and estimate the memory requirements following the method introduced in [**?**], [**?**]. In the inference process, the model needs to allocate memory for the outputs in each layer. The model parameters and output size of each layer of the model are listed and estimated in Table **??**. N in the table is the number of nodes of the input graph. We set $N = 1000$ to estimate the required memory consumption. The space complexity of the model parameters in DeepCas and DeepHawkes is $O(N)$ due to the memory requirements of node representation are related to the total number of nodes, which can be millions. Therefore, the number of model parameters for DeepCas and DeepHawkes is tens of millions. The space complexity of these models, including memory consumption of outputs in each layer is also $O(N)$, which is the same as in Cascade2vec. The estimated memory required for the model parameters of Cascade2vec, DeepCas and DeepHawkes are 340 KB, 260 MB and 280 MB, respectively. If the outputs in each layer are taken into consideration, the memory requirements of these models are 13.8 MB, 266 MB and 288 MB, respectively. When the model is implemented in mini-batched mode, the memory consumption is further increased. When the batch size is set to 32, the memory requirement of Cascade2vec in theory is around 443 MB. In practice, the GPU memory consumption of Cascade2vec is 966MB, which is higher than what is estimated due to cache, temporary variables, biases or framework consumption, etc.

The runtime for an each epoch of Cascade2vec is 35 seconds in the Microblog network dataset at 1 hour. While the runtimes for an epoch of DeepCas and DeepHawkes are 300 seconds and 172 seconds, respectively. However, both

the runtime and the real memory consumption of the models are related to specific implementation. Thus, the comparison is not completely fair since their implementations are based on different frameworks. The preprocessing procedure is also ignored in the complexity analyses. To conclude, our model has a faster runtime in theory and practice than DeepHawkes and DeepCas. However, our work focuses on improving the performance of cascade prediction, not reducing complexities in cascade prediction. The complexity is not our main concern.

VI. CONCLUSION

Cascade prediction is one of the fundamental challenges in social network analysis. The cascades in previous models are learned as a set of random walks or propagation paths without having an effective or favorable graph-level representation. In this paper, we modeled the cascades as dynamic graphs and proposed a new method, Cascade2vec, to learn the representation of cascades by graph recurrent neural networks. To improve the spatial representation of the graphs, we proposed a new graph neural network model that addresses the oversmoothing problems in GNNs and improves the learning ability of graph characteristics. Consequently, the proposed Cascade2vec significantly reduces the mean and median squared log-errors compared with the strong baselines in two cases: retweet prediction in the Microblog network and citation prediction in the APS citation network. Cascade2vec enriches the methods of cascade prediction and proves that a graph neural network is able to perform well in the regression prediction tasks.

In future work, we will take the distributions of the datasets into consideration and design a graph neural network that is more suitable for long-tail distributed data. In addition, external information (e.g., topics and sentiment) will be introduced into the model to enhance the representation of cascades.

REFERENCES

[1] A. Guille, H. Hacid, C. Favre, and D. A. Zighed, "Information diffusion in online social networks: a survey," International Conference on Management of Data, vol. 42, no. 2, pp. 17–28, 2013.

[2] Z. Zhang, C. Liu, X. Zhan, X. Lu, C. Zhang, and Y. Zhang, "Dynamics of information diffusion and its applications on complex networks," Physics Reports, vol. 651, pp. 1–34, 2016.

[3] Z. Zhang, L. Cui, A. Fang, Z. Pan, Z. Zhang, and H. Zhang, "Information dissemination analysis using a time-weight null model: A case study of sina micro-blog," IEEE Access, vol. 6, pp. 71 181–71 193, 2018.

[4] M. Jenders, G. Kasneci, and F. Naumann, "Analyzing and predicting viral tweets," Proceedings of the 22nd International Conference on World Wide Web companion. ACM, pp. 657–664, 2013.

[5] J. Cheng, L. A. Adamic, P. A. Dow, J. M. Kleinberg, and J. Leskovec, "Can cascades be predicted," Proceedings of the 23rd International Conference on World Wide Web 2014, pp. 925–936, 2014.

[6] V. D. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," Journal of Statistical Mechanics: Theory and Experiment, vol. 2008, no. 10, p. 10008, 2008.

[7] H. Pinto, J. M. Almeida, and M. A. Goncalves, "Using early view patterns to predict the popularity of youtube videos," Proceedings of the Sixth ACM International Conference on Web Search and Data Mining (WSDM '13), pp. 365–374, 2013.

[8] Q. Zhao, M. A. Erdogdu, H. Y. He, A. Rajaraman, and J. Leskovec, "Seismic: A self-exciting point process model for predicting tweet popularity," Proceedings of ACM SIGKDD international conference on Knowledge Discovery and Data Mining, pp. 1513–1522, 2015.

[9] Q. Cao, H. Shen, H. Gao, J. Gao, and X. Cheng, "Predicting the popularity of online content with group-specific models," Proceedings of the 26th Conference Companion on International Conference on World Wide Web, pp. 765–766, 2017.

[10] C. Li, J. Ma, X. Guo, and Q. Mei, "Deepcas: An end-to-end predictor of information cascades," in Proceedings of the 26th International Conference on World Wide Web, 2017, pp. 577–586.

[11] Q. Cao, H. Shen, K. Cen, W. Ouyang, and X. Cheng, "Deephawkes: Bridging the gap between prediction and understanding of information cascades," in Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. ACM, 2017, pp. 1149–1158.

[12] C. Gou, H. Shen, P. Du, D. Wu, Y. Liu, and X. Cheng, "Learning sequential features for cascade outbreak prediction," Knowledge and Information Systems, vol. 57, no. 3, pp. 721–739, 2018.

[13] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, and M. Sun, "Graph neural networks: A review of methods and applications." arXiv: Learning, 2018.

[14] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," 2016, pp. 3844–3852.

[15] T. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," International Conference on Learning Representations, 2017.

[16] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," International Conference on Learning Representations, 2018.

[17] Z. Muhan, C. Zhicheng, N. Marion, and C. Yixin, "An end-to-end deep learning architecture for graph classification," AAAI Conference on Artificial Intelligence, pp. 4438–4445, 2018.

[18] F. Chen, S. Pan, J. Jiang, H. Huo, and G. Long, "Dagcn: Dual attention graph convolutional networks," arXiv: Learning, 2019.

[19] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" International Conference on Learning Representations, 2018.

[20] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Gated feedback recurrent neural networks," International Conference on Machine Learning, pp. 2067–2075, 2015.

[21] O. Tsur and A. Rappoport, "What's in a hashtag?: content based prediction of the spread of ideas in microblogging communities," Proceedings of the Fifth ACM International Conference on Web Search and Data Mining (WSDM '12), pp. 643–652, 2012.

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

[23] S. Goel, A. Anderson, J. M. Hofman, and D. J. Watts, "The structural virality of online diffusion," Management Science, vol. 62, no. 1, pp. 180–196, 2015.

[24] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks." arXiv: Learning, 2019.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," Computer Vision and Pattern Recognition, pp. 770–778, 2016.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," Neural Information Processing Systems, pp. 5998–6008, 2017.

[27] L. Hong, O. Dan, and B. D. Davison, "Predicting popular messages in twitter," International Conference on World Wide Web Companion, pp. 57–58, 2011.

[28] Z. Ma, A. Sun, and G. Cong, "On predicting the popularity of newly emerging hashtags in twitter," Journal of the Association for Information Science and Technology, vol. 64, no. 7, pp. 1399–1410, 2013.

[29] P. Bao, H. Shen, J. Huang, and X. Cheng, "Popularity prediction in microblogging network: a case study on sina weibo," International Conference on World Wide Web Companion, pp. 177–178, 2013.

[30] L. Weng, F. Menczer, and Y. Ahn, "Virality prediction and community structure in social networks," Scientific Reports, vol. 3, no. 1, pp. 2522–2522, 2013.

[31] C. Yi, Y. Bao, and Y. Xue, "Mining the key predictors for event outbreaks in social networks," Physica A-statistical Mechanics and Its Applications, vol. 447, pp. 247–260, 2016.

[32] O. Tsur and A. Rappoport, "What's in a hashtag?: content based prediction of the spread of ideas in microblogging communities," International Conference on Web Search and Data Mining, pp. 643–652, 2012.
[33] H. Shen, D. Wang, C. Song, and A. Barabasi, "Modeling and predicting popularity dynamics via reinforced poisson processes," National Conference on Artificial Intelligence, pp. 291–297, 2014.
[34] P. Bao, H. Shen, X. Jin, and X. Cheng, "Modeling and predicting popularity dynamics of microblogs using self-excited hawkes processes," International Conference on World Wide Web Companion, pp. 9–10, 2015.
[35] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," IEEE Transactions on Neural Networks, vol. 20, no. 1, pp. 61–80, 2009.
[36] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," Applied and Computational Harmonic Analysis, vol. 30, no. 2, pp. 129–150, 2011.
[37] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," International Conference on Learning Representations, 2018.
[38] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," International Joint Conference on Artificial Intelligence, pp. 3634–3640, 2018.
[39] R. Zafarani, M. A. Abbasi, and H. Liu, Social Media Mining, 2015.
[40] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," International Conference on Machine Learning, pp. 448–456, 2015.
[41] P. Yanardag and S. V. N. Vishwanathan, "Deep graph kernels," Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1365–1374, 2015.
[42] F. R. K. Chung, "The diameter and laplacian eigenvalues of directed graphs," Electronic Journal of Combinatorics, vol. 13, no. 1, p. 4, 2006.
[43] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," Applied and Computational Harmonic Analysis, vol. 30, no. 2, pp. 129–150, 2011.
[44] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," COURSERA: Neural networks for machine learning, vol. 4, no. 2, pp. 26–31, 2012.
[45] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," Neural Information Processing Systems, pp. 3111–3119, 2013.
[46] Z. Huang, Z. Wang, Y. Zhu, C. Yi, and T. Su, "Prediction of cascade structure and outbreaks recurrence in microblogs," Social Media Processing. SMP 2017. Communications in Computer and Information Science., pp. 53–64, 2017.
[47] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.
[48] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to algorithms. MIT press, 2009.
[49] E. Tsironi, P. Barros, C. Weber, and S. Wermter, "An analysis of convolutional long short-term memory recurrent neural networks for gesture recognition," Neurocomputing, vol. 268, pp. 76–86, 2017.
[50] K. He and J. Sun, "Convolutional neural networks at constrained time cost," Computer Vision and Pattern Recognition, pp. 5353–5360, 2015.
[51] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," Computer Vision and Pattern Recognition, pp. 2818–2826, 2016.

ZHENHUA HUANG was born in Anhui, China. He received the B.S. degree in School of Software Engineering from South China University of Technology in 2014. He is currently a Ph.D Candidate in the School of Software Engineering at South China University of Technology, Guangzhou. He was a visiting scholar and young big data scientist worked at University of California, Irvine in the program of IBM-CSC Y-100. He has authored more than five articles. His research interests include social computing, sentiment analysis, and deep learning.

ZHENYU WANG received the Ph.D. degree from department of computer science, Harbin Institute of Technology in 1993. He is the director of the Chinese Information Community of China, director of the Guangdong Provincial Social Media Processing and Engineering Center. His research interests include natural language processing, text mining, and social network analysis. He has more than 80 papers published. Received the IBM-Ministry of Education University Cooperation Project Excellent Teacher Awards several times, guiding students to win the first and second prizes of the IBM National Competition.

RUI ZHANG received the B.S. degree in School of Software Engineering from South China University of Technology, Guangzhou, China in 2015. He has been a Ph.D. candidate in School of Software Engineering at South China University of Technology since 2015. His research interests include natural language generation, text mining and sentiment analysis.