DeepGraph: Graph Structure Predicts Egonet Growth

Cheng Li[1], Xiaoxiao Guo[2], Qiaozhu Mei[1,2]
[1]School of Information, University of Michigan, Ann Arbor, MI, USA
[2]Department of EECS, University of Michigan, Ann Arbor, MI, USA
{lichengz, guoxiao, qmei}@umich.edu

ABSTRACT

The topological (or graph) structure of one's ego network is known to be predictive of multiple dynamic properties of the ego center. For instance, a researcher's collaboration network is predictive of her future h-index. Conventionally, a graph structure is represented using an adjacency matrix or a set of hand-crafted structural features. These representations either fail to highlight local and global properties of the graph or suffer from a severe loss of structural information. There lacks an effective graph representation, which hinges the realization of the predictive power of network structures.

In this study, we propose to learn the representation of the topological structure of a egonet through a deep learning model. This end-to-end prediction model, named DeepGraph, takes the input of the raw adjacency matrix of a egonet and outputs a prediction of the growth of the network. The adjacency matrix is first represented using a graph descriptor based on the heat kernel signature, which is then passed through a multi-column, multi-resolution convolutional neural network. Extensive experiments on four large collections of real-world networks demonstrate that the proposed prediction model significantly improves the effectiveness of existing methods, including linear or nonlinear regressors that use hand-crafted features, graph kernels, and competing deep learning methods.

1. INTRODUCTION

Today we are surrounded by real-world networks of people, information, and technology. These heterogeneous, large scale, and fast evolving networks have provided a new perspective of scientific research, which has resulted in a rapid development of new theories, algorithms, and applications.

How to model and predict the dynamic properties of social or information networks has received considerable attentions recently [33, 42, 1, 27, 20, 35, 8]. In the present work, we focus specifically on k-hop egonets, which are composed of a "ego" node, and its k-hop neighbors. Many interesting properties could be studied from these ego networks, including the size of the network, metrics of individual nodes or structures (e.g., degree or diameter), or even external properties that are not directly observed from the network structure (e.g., prestige, productivity or revenue of the ego center). All these properties change over time, and their dynamics can be generally referred to as the *growth* of a ego network[1]. Indeed, the prestige of an individual node grows with the size of its egonet. Accurate prediction of network growth has many valuable applications. For example, predicting the growth of paper's citation networks helps scientists to identify promising research directions; predicting the growth of Facebook user's friendship networks helps social network vendors optimize their marketing strategies.

Taking a typical data mining perspective, most existing methods extract features from both the network itself and any external information sources available. A function is learned that takes these features as input and outputs a predicted value of the network property in the future [1]. From many explorations on different genres of networks, there has been a consensus in literature that features extracted from the topological structure of the network (a.k.a., the *graph*) are generally very informative in these prediction tasks [1, 8]. As a comparison, other types of information, e.g., content or demographics, are only useful in certain scenarios. For example, the content of a hashtag is predictive to its diffusion [43] and homophily (e.g., similar demongraphics) is predictive to the growth of friendship networks [7], but these effects are not generalizable to other networks and other dynamic properties. In this study, we focus on investigating the predictive power of the *graph structure* of a ego network on its growth.

Existing structural features are typically hand-crafted based on theoretical and empirical findings in the social network literature. For example, open triads with two strong ties are likely to be closed in the near future [11]; dense communities are resistant to novel information and they grow slower than others [16]; nodes spanning structural holes are likely to gain social capital and experience a rapid growth of its prestige and other properties [6]. Features such as network density, clustering coefficients, triadic profiles, and structural holes are therefore designed to implement these intuitions and represent the graph structure.

Despite the success in predicting network growth, there are observable issues of representing the topological structure of a network using these hand-crafted features. Some of them only describe a global property of the network, such as network density or degree distribution; some of them provide a fine-grained description of local structures but fail to capture global information, such as triads and other substructures; others lie between the two extremes, such as structural holes. None of these features is able to fully represent both the local and the global structure of a graph and the complex interaction between local and global properties.

[1]The *growth* refers to both the increment and decrement of the dynamic properties of the ego networks, i.e., positive or negative *growth*.

On the other hand, these heuristic features usually have a limited characterization power of networks, as many networks may share the same feature representation. For example, most real-world networks at scale may have a similar (power-law) degree distribution, and two very different networks may happen to have the same ratio of closed triangles. Taking a machine-learning point of view, we are intrigued by the following questions: *what is a suitable representation of network structure and how effective is such a representation when used to predict network growth?*

Our answers to the two questions are inspired by the recent developments in deep learning and graph representation. We introduce a graph descriptor that is based on the Heat Kernel Signature (HKS) [31], which serves as a universal low-level representation of the topological structures of networks. HKS has been successfully employed in representing the surface of 3D objects [13, 39]. By modeling the amount of heat flow over nodes of a network over time, HKS successfully stores both the global and the local structural information of the entire network. Using a histogram to describe the probability distribution of heat values at a series of time points [13, 39], isomorphic networks (networks with the same topological structure) can be mapped to a unique representation at little loss of structural information. However, unlike 3D objects which are composed of polygon meshes, the structures of networks vary in shape, size, and complex local structures. To address this issue, some computations of HKS need to be approximated carefully. Inspired by the semantics of the HKS-based graph descriptors, we propose a multicolumn, multiresolution neural network that learns latent hierarchical representations of graphs on top of the HKS-based graph descriptor. The proposed deep neural network, named DeepGraph, predicts network growth in an end-to-end process.

We conduct extensive experiments to evaluate the effectiveness of DeepGraph. Different growing properties are predicted for four genres of real-world ego networks. Empirical results show that our method outperforms baseline approaches that use alternative graph representations, hand-crafted features, or existing deep learning architectures. High-level representations learned by DeepGraph well connect to existing findings in the social network literature.

2. RELATED WORK

Predicting the growth of networks or the evolution of certain properties of networks has been widely studied. People attempt to predict the dynamics of various network metrics or aggregated activities in a network, e.g., the number of up-votes on Digg stories [33], the number of newly infected nodes in diffusion [42], the growth of a community [1, 27], or the dynamics of a cascade [20, 35, 8]. In these studies, a set of problem-specific features are usually manually designed based on the network structure, textual content, user demographics, historical statistics, and other sources of information. Among them, the features extracted from the network structure are both effective in individual tasks and robust across different tasks. In this work, we limit our focus on information purely from the network structure.

Finding a suitable representation of the topological structure of a network has always been a critical preliminary step of network analysis. Conventionally, a network is represented as an adjacency matrix or a sparse list of edges. However, these lossless representations do not effectively present the structural characteristics of the network. Moreover, they are sensitive to the manipulation of node orders, making networks with the same topological structure mapped to different representations. Other approaches represent the network structure with a series of network metrics and/or a set of structural patterns (e.g., triads [19], quads [37], or meta-paths [32]). Arbitrary higher-order substructures can be included, such as communities and structural holes. These bag-of-substructures better capture local patterns of the network structure.The major problem of this approach is that it is computationally infeasible to enumerate high-order substructures, and low-level substructures have limited representation power of the global structure of the network. As a result, many different networks may share the same or similar bag-of-substructures.

In graph classification, a myriad of graph kernel methods are proposed which compute pairwise similarities between graphs [17, 2, 30, 29]. For example, *graphlets* [29, 36] computes the graph similarity based on the distribution of induced, non-isomorphic subgraphs. Some other graph kernels integrate frequent graph mining into the model training process [28, 26]. Graph kernels provide an indirect representation of networks so that similar structured networks yield a high value through the graph kernel function. The burden of graph kernels is the design of effective kernels. In the paper, we compare existing graph kernels to highlight the flexibility of our model.

Recently, researchers have started to apply deep learning to network structure representation learning. Several proposals have been made to learn a low-dimensional vector representation of individual nodes by considering their neighborhood [34, 25, 15]. Deep learning techniques have also improved graph kernels for graph structure learning[40, 41, 23]. Recently, Niepert et al. [24] applied convolution over receptive fields constructed by sequence of neighboring nodes. These methods focus only on the local structure of a graph and graph kernels require expensive pairwise comparisons. In the paper, we compare our model to these alternative deep learning approaches and show the performance advantage of our model.

Heat kernels have been studied for the task of graph clustering [3], graph partitioning [12], and modeling social network marketing processes [21]. These applications rely on the raw output of heat kernels for a variety of tasks, rather than developing a *signature*, nor do they abstract representation of graphs base on heat kernels. In the community of computer vision, Heat kernel signature has been successfully used to model 3D objects [31, 13, 39], whose surfaces are defined by polygon meshes, a network composed of simple convex polygons. In contrast, real-world networks are consist of various shapes, sizes, and local structures. How to represent arbitary networks with heat kernel signatures and how to predict network growth using such a signature remain a challenging question to be studied.

3. DEEPGRAPH FOR NETWORK GROWTH PREDICTION

We propose a unified predictive neural network model to learn graph structure representation for network growth prediction problem. The proposed predictive model, named DeepGraph, combines heat kernel signature and deep neural networks. Below we describe the two key components of our model, (1) a heat kernel signature based graph descriptor and (2) a deep multi-column, multi-resolution convolutional neural network, in turn, following a brief definition of the network growth prediction problem.

3.1 Problem Formulation and Notations

Given a real-world network snapshot at time t, denote its graph structure as $\mathcal{G}^{(t)} = (V, E)$, with a set of nodes V and a set of edges E. A node $i \in V$ represents an entity (e.g., an actor in a social network or a paper in a citation network), an edge $(i, j) \in E$ represents a relationship (e.g., friendship, citation, or influence) between node i and node j. An adjacency matrix $\mathbf{W} \in \mathbb{R}^{|V| \times |V|}$ encodes the topological structure of the graph $\mathcal{G}$. In this work, we consider the binary adjacency matrix. Its element w_{ij} is 1 if and only if $(i,j) \in E$ and 0 otherwise.

A network property is a function that maps a graph structure $\mathcal{G}^{(t)}$ to a property value $y^{(t)} \in \mathbb{R}$. For example, a network property could be the number of friends given a user's Facebook ego-network. A network growth predictor is a function that maps a graph structure $\mathcal{G}^{(t)}$ to a property value $y^{(t')}$ at time t', satisfying $t' > t$. For example, a network growth predictor could map a user's Facebook ego-network of this year to the number of friends next year.

The network growth prediction can be naturally formulated as a supervised learning problem. Specifically, the problem is to derive a network growth predictor f given a training set of tuples $\{(\mathcal{G}_i^{(t_i)}, y_i^{(t'_i)})\}_{i=1}^{M}$ to minimize the prediction error over a test set of tuples $\{(\mathcal{G}_j^{(t_j)}, y_j^{(t'_j)})\}_{j=1}^{N}$ satisfying $\forall i\ t'_i > t_i$, $\forall j\ t'_j > t_j$, $\min_j(t_j) > \max_i(t_i)$, and $\min_j(t'_j) > \max_i(t'_i)$. The time ordering constraints highlights the practical motivation that we are interested in using historical data to predict future properties of current networks.[2] To apply a machine learning algorithm, it is critical to first represent the graph $\mathcal{G}^{(t)}$ computationally, such as using a vector of features.

3.2 Heat Kernel Signature based Graph Descriptors

The motivation in adopting Heat Kernel Signature (HKS) is its theoretical proven properties in representing graphs: HKS is an **intrinsic** and **informative** representation for graphs [31]. **Intrinsicness** means that isomorphic graphs map to the same HKS representation, and **informativeness** means if two graphs have the same HKS representation, then they must be isomorphic graphs. Our HKS-based graph descriptor builds on the theoretical properties of HKS and further provides universal representations for graph with different sizes in network growth prediction.

Heat kernel function. Formally, the heat kernel $h_z(i,j)$, a function of two nodes i, j at any given diffusion step z, denotes the amount of aggregated heat flow through all edges among two nodes after diffusion step z[3]. In computer vision, graphs are stored as meshed networks and heat kernels are computed by finding eigenfunctions of the Laplace-Beltrami operator [31]. However, meshed networks are not available for most real-world networks. Instead, we use eigenfunction expansion of a graph Laplacian [31, 3] to compute the heat kernel for information networks. Given a graph $\mathcal{G} = (V, E, W)$, the graph Laplacian is defined as: $\mathbf{L} = \mathbf{D} - \mathbf{W}$, where D is a diagonal degree matrix with diagonal entries being the summation of rows of W: $D_{ii} = \sum_j w_{ij}$. The normalized Laplacian of the graph is given by $\mathbf{L_N} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}$.

The heat kernel is then defined as

$$h_z(i,j) = \sum_{k=1}^{|V|} e^{-\lambda_k z}\phi_k(i)\phi_k(j) \tag{1}$$

where λ_k is the k-th eigenvalue of the normalized Laplacian $\mathbf{L_N}$ and ϕ_k is the k-th eigenfunction s.t. $\sum_i |\phi_k(i)|^2 = 1$. Note that the eigenvalues might be unreal in the case of directed graphs. There has been studies on how to tackle this problem [9]. In this work, for simplicity, we convert directed graphs to undirected ones by applying $\mathbf{W} = (\mathbf{W} + \mathbf{W}^\intercal)/2$.

Heat kernel signature. Heat kernel signature was introduced to mitigate the computation bottleneck of using heat kernel functions in representing graphs. Both heat kernel and heat kernel signature are proven to be intrinsic and stable against noises. However, the computation complexity of using heat kernel as a point signature is overwhelming since the point signature, $\{k_t(v,.)\}_{t>0}$, is defined on the product of temporal and spatial domain. Heat kernel signature simplifies the computation by considering only a subset of product of temporal and spatial domain while keeping as much information as possible. Specifically, heat kernel signature reduces the computation complexity by only requiring $h_z(v,v)$ over a finite set of N diffusion steps $z \in \{z_1, z_2, ..., z_N\}$ for $\forall v \in V$ without losing the intrinsic and informative properties.

Formally, a heat kernel signature (HKS) is a matrix $\mathbf{H} \in \mathbb{R}^{|V| \times N}$ satisfying

$$H_{ij} = h_{z_j}(i,i) \tag{2}$$

These time points are sampled with equal difference after logarithm [31], such that $\log z_n - \log z_{n-1} = \log z_{n+1} - \log z_n$.

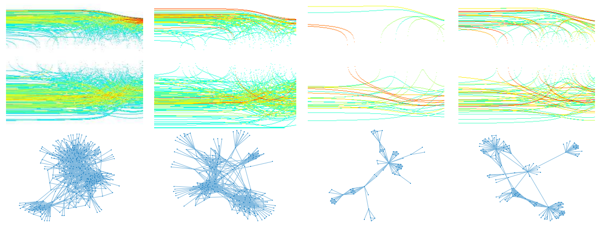

Figure 1: Examples of HKS-based graph descriptors. The first row shows our graph descriptors for graphs in the second row. Figure (a) and (b) are subnetworks from Facebook [38]. Figure (c) and (d) are some authors' collaboration networks built from ACL Anthology [10].

Graph descriptor. The practical issues in combining HKS and deep neural networks are that we need a global vertex indexing to guarantee the uniqueness and that the size depends on $|V|$. We further process heat kernel signature $\mathbf{H}$ into a universal representation independent of $|V|$ using a histogram conversion. Specifically, we use histograms to estimate the distribution of HKS values in each column[4]. By denoting N_B the number of bins used in the histogram, we obtain a universal descriptor $\mathbf{S} \in \mathbb{R}^{N_B \times N}$. Unlike HKS, the new descriptor is independent of vertex ordering and vertex number. We call this final matrix *graph descriptor*, $\mathbf{S}(\mathcal{G})$, as it is adapted to describe information networks. Figure 1 shows four examples of our graph descriptors for real world graph structures.

Graph descriptor vs. adjacency matrix. We have described the process in converting an adjacency matrix into our graph descriptor, which is then passed through a deep neural network for further feature extraction. All computation in this process is to obtain a more effective low-level representation of the topological structure information than the original adjacency matrix.

First, isometric graphs could be represented by many different adjacency matrices, while our graph descriptor would provide a unique representation for those isometric graphs. The unique representation simplifies the neural network structures for network growth prediction.

[2]In practice, researchers focus on a specially case of the network growth prediction problem with the equal interval increment constraint, $t'_j - t_j = t'_i - t_i = C > 0$ [20, 35].

[3]The diffusion is simulated for a given graph snapshot. The heat kernel computation does not require graph snapshot at other timestamps. The diffusion step z should not be confused with the network timestamp t.

[4]The bin ranges are aligned column-wise on the training data.

Second, our graph descriptor provides similar representations for graphs with similar structures. The similarity of graphs is less preserved in adjacency matrix representation. Such information loss could cause great burden for deep neural networks in growth prediction tasks.

Third, our graph descriptor is a universal graph structure representation which does not depend on vertex ordering or the number of vertexes, while the adjacency matrix is not.

Time complexity. The major overhead of computing graph descriptors lies in the calculation of eigenvectors. The time complexity of computing eigenvectors is $\mathcal{O}(K|V|^2)$ where K is the number of eigenvectors. Our graph descriptors finish in acceptable time frame for real world network data. The data description and time complexity analysis are in Section 4.

Semantics of graph descriptor. The rows and columns in our graph descriptor reflect the network topology from different perspectives. The rows express the heat density dynamics over diffusion steps, and the columns capture the static heat density patterns for a given diffusion step. Successive rows or columns express higher-order properties of the topology structure information. Such representational properties motivate the adoption of row-wise and column-wise convolution networks for feature learning.

3.3 Deep Graph Descriptor

As information abounds in the raw representation extracted by the HKS-based graph descriptor, applying a simple regressor, e.g., linear regression, could fail to fully extract useful information from it. In contrast, deep neural networks (DNN) have achieved tremendous success in learning latent representations from raw inputs in a compositional hierarchy. Combining DNN and HKS-based graph descriptor together thus offers an opportunity to address the graph structure representation challenges in predicting network growth. Inspired by the semantics of the graph descriptors, we propose a deep multicolumn, multiresolution convolutional neural networks for the network growth prediction task.

Multiresolution convolutions. Our model builds on the multiresolution 1-D convolution (MrConv) which maps an input matrix into a feature map matrix. Specifically, let $\mathbf{x}_i \in \mathbb{R}^k$ denote the i-th row of the input matrix. The input is then represented as $\mathbf{x}_{1:n} = \oplus_{i=1}^{n} \mathbf{x}_i$ where $\oplus$ is the concatenation operator and n is the number of rows. The 1-D convolution with a filter size m apply a filter $\mathbf{w} \in \mathbb{R}^{mk}$ to each possible window of m rows to produce a new feature vector $\mathbf{c} = [c_1, c_2, ..., c_{n-m+1}]$. The feature c_i is generated from a window of m rows $\mathbf{x}_{i:i+m-1}$ by $c_i = g(\mathbf{w}^\intercal \mathbf{x}_{i:i+m-1} + b)$, where $b \in \mathbb{R}$ is a bias term and g is a non-linear function such as a hyperbolic tangent function or a rectified non-linearity function.

We have described the process by which *one* feature vector $\mathbf{c}$ is extracted from *one* filter. Our multiresolution convolution (MrConv) layer uses multiple filters with varying filter sizes to obtain multiple resolution features. Specifically, one MrConv layer has l different convolution filter sizes $\{m_1, m_2, ..., m_l\}$. The filter of size m generates a corresponding feature vector $\mathbf{c}^{(m)}$. Feature vectors generated by different filter sizes are then concatenated into one vector $\mathbf{c}^* = \oplus_{i=1}^{l} \mathbf{c}^{(m_i)}$. Moreover, we extend each filter size to have d different filters. The final output feature map is a matrix $\mathbf{O}$ where each column is a feature vector $\mathbf{c}^*$ and there are d columns: $\mathbf{O} = \left(\mathbf{c}^{*1}, \mathbf{c}^{*2}, ..., \mathbf{c}^{*d}\right)$.

An example of our MrConv is shown in Figure 2(a). The example MrConv layer has two different filter sizes $\{1, 2\}$. Each filter size has three different filters, whose feature vectors form different columns in the final feature map. Multiple multiresolution convolution layers are stacked to form our model.

Multicolumn model. Inspired by the different semantics of rows and columns in the HKS-based graph descriptor, our model deploys a two network-column structure, as shown in Figure 2(b). One column uses multiresolution 1-D convolution (MrConv) operations over the graph descriptor bins and the other one uses MrConv over diffusion times. The two columns extract different features from the graph descriptors at multiple resolution scales. Intuitively, the first column extracts statistical features of the density dynamics in diffusion. The second column extracts features on static density pattern for different diffusion steps. Both kinds of features reflect the topology of the underlying graph structure, but explain the structure topology from different perspectives. A single column convolutional neural network can hardly extract such two kinds of features successfully.

The feature maps from the two columns are then concatenated and passed through multiple dense (i.e. fully-connected) layers with non-linear activation functions. The output from the multiple dense layers are then passed through a final linear fully-connected layer with only one output unit. The output unit $\hat{y}$ is thus the network growth prediction of our model.

3.4 End-to-End Training

Let McMrConv$(., \theta)$ denote the multicolumn multiresolution convolutional neural network with parameters θ. The final output of our neural network given a graph $\mathcal{G}_k$ is represented as: $\hat{y}_k =$ McMrConv$(\mathbf{S}(\mathcal{G}_k), \theta)$. Given a training data set $\{(\mathcal{G}_k, y_k)\}_{k=1}^{K}$, the deep neural network is trained to minimize the average squared error: $\mathcal{L}(\theta) = \frac{1}{K}\sum_{k=1}^{K}\Big(\text{McMrConv}(\mathbf{S}(\mathcal{G}_k), \theta) - y_k\Big)^2$. The HKS-based graph descriptor and the deep neural network assembles DeepGraph, an end-to-end deep architecture to predict network growth based on graph structure.

4. EXPERIMENT SETUP

We compare our model with existing approaches on the network growth prediction problem. We then evaluate variants of our model for credit assignment.

4.1 Data sets

When selecting real-world data sets for evaluation, we consider both popularity and diversity of the application scenarios. The four data sets we choose include ego networks extracted from social networks, scientific collaboration networks, and entertainment networks. The statistics of these data sets are presented in Table 1. Please note that due to the diverse nature of the data sets and the various precision of timestamps available, it is hard to apply an unified time frame for all data sets. Viewed from another perspective, this helps us evaluate the flexibility and generality of our methods, verifying whether it can be applied to any length and granularity of time frames.

We follow the procedure described in [40] to construct ego-nets. The Facebook data set is collected from the New Orleans networks [38], where nodes are Facebook users and edges are friendships. We derive the snapshot of ego-networks for each user according to the timestamps listed in Table 1, which is used to predict the number of new friends this user made in the next four months.

As the YouTube [22] data set also describes user friendships, it follows the same setting as Facebook.

The AAN data set [10] is built upon scientific publications from the ACL Anthology[5], where nodes are authors and edges are col-

[5]`http://aclweb.org/anthology/`

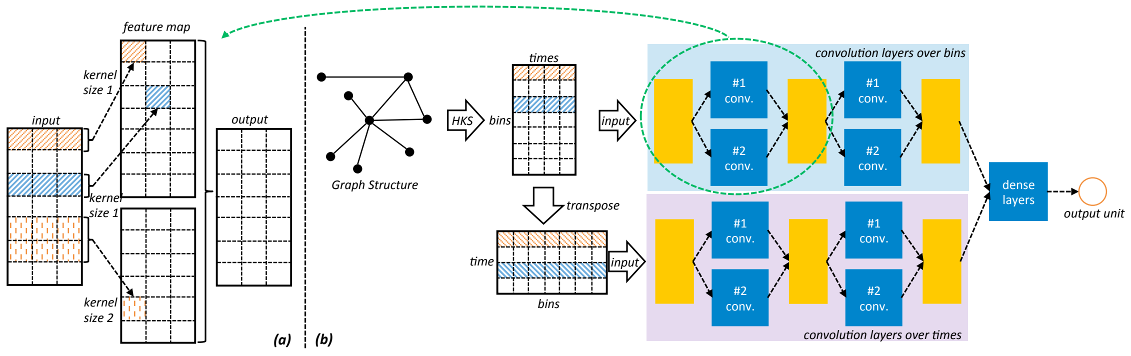

Figure 2: (a) An example of the multiresolution convolution unit with two kernel sizes. (b) An example of the multicolumn, multiresolution deep neural network model for network growth prediction with two convolution layers.

Table 1: Statistics of the data sets.

	Dataset	Facebook	YouTube	AAN	IMDB
# graphs	train	12990	15258	8426	12500
	val	890	1283	713	1017
	test	2092	3273	1722	2407
Avg. nodes	train	399.9	147.6	271.4	197.3
	val	397.5	167.3	302.4	208
	test	436	165.8	402.4	216
Avg. edges	train	6800.8	1439.2	2079.8	7801.5
	val	6764.4	1626.1	2327.2	7847.7
	test	7499.2	1620.9	3321.8	7964.5
Avg. growth	train	3.6	9	1.2	1.3
	val	3.8	10.8	1.2	1.3
	test	3.4	9.3	1.2	1.3
Avg. scaled growth[1]	train	1.7	2.4	0.9	1
	val	1.8	2.2	0.9	1
	test	1.6	2.1	0.9	0.9
Graph time[2]	train	2007.6	2007.2	2009	2000
	val	2007.7	2007.3	2010	2001
	test	2007.8	2007.4	2011	2002
Growth time[3]	train	2007.10	2007.4	2010	2001
	val	2007.11	2007.5	2011	2002
	test	2007.12	2007.6	2012	2003
k-hop ego-net[4]	all	2	2	3	2

1. *Avg. scaled growth* scales label y to $\log_2(y+1)$ [20, 35].
2. *Graph time* of 2007.6 means the graph is built by taking the snapshot of Jun. 1, 2007.
3. *Growth time* of 2007.10 means the growth is computed between its corresponding *graph time* to Oct. 1, 2007. Graphs in train/val/test set do not overlap.
4. *k-hop ego-net* for AAN is set to 3, due to its small size when $k = 2$.

laboration. Each author's ego-nets are extracted to predict her h-index in the next year.

IMDB is a movie co-star data set[6], where nodes are actors or actresses, and an edge is formed if they appear in the same movie. The ego-nets of each actor/actress is used to predict the number of new movies the actor/actress produced in the next year.

To examine whether we can truly predict future growth, we make sure of two important points: (1) the period to compute growth for test set is always later than that for training set; (2) one graph can only appear in one of the training, validation and test set. To this end, for each node in the global network, they are randomly assigned to the training/validation/test set with probability of 0.8/0.05/0.15. Based on which set they are in, their ego-nets and growth are computed according to the time listed in Table 1. If a node has not yet been created for the given time, it is simply removed.

We notice that the growth of all the ego networks in general follows a power-law distribution, where a large number of networks did not grow at all. Therefore we downsampled 50% graphs of each train/val/test set with zero growth (to the numbers shown in Table 1) and applied a logarithm transformation of the outcome variable (network growth), following [20, 35]. The network growth are scaled logarithmically for two reasons. First, baseline methods with linear regression are sensitive to extremely large outcomes. Second, when a network grows to a considerably large scale, we care more about its scale rather than the exact number.

4.2 Evaluation Metric

We use mean squared error (MSE) as our evaluation metric, which is a common choice for regression tasks. Specifically, denote $\hat{y}$ a prediction value, and y the ground truth value, then MSE $= \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$. As noted before, y in above equation is a scaled version of the original value y^o, that is $y = \log_2(y^o + 1)$.

4.3 Baseline methods

We compare DeepGraph with methods from two categories: feature-based methods used for network prediction tasks, and alternative graph representation methods.

Feature based. Many structural features have been designed for various network prediction tasks [1, 27, 35, 8]. We select from them those that could be generalized across data sets, including:

Frequencies of k-node substructures ($k \leq 4$)[37]. This counts the number of nodes ($k = 1$), edges ($k = 2$), triads (e.g., the number of closed and open triangles) and quads.

Other network properties: average degree, the length of the shortest path, edge density, the number of leaf nodes (nodes with degree 1), the number of leaf edges, the average closeness of all nodes, clustering coefficient, diameter, and the number of communities obtained by a community detection algorithm [4].

Graph kernels. Following [24], we compare with four state-of-the-art graph kernels: the shortest-path kernel (**SP**) [5], the random walk kernel (**RW**) [14], the graphlet count kernel (**GK**) [29], and the Weisfeiler-Lehman subtree kernel (**WL**) [30]. In our experiment, the RW kernel does not finish after 10 days for a single data set, so we exclude it for comparison. This exclusion is also observed for the same reason in [24, 41].

-linear and **-deep**. Feature based methods and graph kernels are usually trained on SVMs. We report linear regression instead, as

[6]`http://www.imdb.com/`

SVM empirically generates poor results for our regression tasks. We append **-linear** to each method to indicate usage of linear regression. To obtain even stronger baselines, we apply deep learning to both feature vectors and graph kernels, indicated by **-deep**.

Smoothed graph kernels. Yanardag et al. [41] apply smoothing to graph kernels, which extends their method of deep graph kernels [40] by considering structural similarity between sub-structures. We report smoothed results only on deep neural networks as it outperforms alternatives empirically.

PSCN, which applies convolutional neural networks (CNN) to locally connected regions from graphs [24], achieving better results over graph kernels on some of the classification data sets.

Hyper-parameters. All hyper-parameters are tuned to obtain the best results on validation set. For linear regression, we chose the L2-coefficient from $\{10^0, 10^{-1}, ..., 10^{-7}\}$. For neural network regression, the initial learning rate is selected from {0.1, 0.05, 0.01, ..., 10^{-4}}, the number of hidden layers from $\{1, 2, ..., 4\}$, and the hidden layer size from $\{32, 64, ..., 1024\}$. The size of the graphlets for GK is chosen from $\{3, 4\}$ (higher than 4 is extremely slow), the height parameter of WL from $\{2, 3, 4\}$, the discount parameter for smoothed graph kernels from $\{1, 0.8, ..., 0\}$. Following [24] for PSCN, the width is set to the average number of nodes, and the receptive field size is chosen between 5 and 10.

Notes. Please notice that in our experiments we are not identifying the nodes in the networks or using the information of the nodes outside the network itself. Of course, knowing the president of United States is in the network provides more confidence on its growth. We choose not to identify nodes because (1) this study focuses on investigating the predictive power of the topological structure of networks, and (2) in practice information about individual nodes may not be available for privacy reasons. For the same reasons, we do not include any information other than the network structure (e.g., content of tweets, or historical metrics of the network) in the prediction task, even though including more information may improve the prediction accuracy.

4.4 DeepGraph Model Parameters

Parameters included in HKS are set to default values across all data sets without further tuning. In Equation 2, we set $t_1 = 0.1$, $t_N = 25$, and $N = 64$. Number of bins N_B is set to 64. To compute histograms, HKS values above $+1.2$ and below -1.2 standard deviation are respectively put to the first and last bins. Values in between are assigned to the remaining equally divided 62 bins.

We perform standard normalization for the histograms of graphs. Each histogram is preprocessed by pixel-wise normalization. We compute the mean and standard deviation for each pixel over the training data set. Then each pixel is normalized by subtracting the corresponding mean value and being divided by sd[7] .

We initialize the parameters of the neural networks using a Gaussian distribution with zero mean and unit standard deviation. An adaptive optimizer, Adam, is used to optimize the parameters of the neural networks. Default hyper-parameters of Adam are used [18].

Structure related hyper-parameters of DeepGraph is set to be the same across datasets. There are two multiresolution convolution layers for each network column, with number of filters 32 and 16. For each convolution layer, we apply three sizes of filters, which are 2, 4, and 6. TanH is used as the activation function. There are two fully connected layers both of size 256. Dropout is applied to the last two dense layers with probability of 0.5. Other learning parameters are listed in Table 2.

4.5 Variants of DeepGraph

[7] $\epsilon = 10^{-8}$ is added to the denominator to avoid numeric issues.

Table 2: Setup of hyper-parameters for DeepGraph.

	Facebook	YouTube	AAN	IMDB
L2-coefficient	1e-5	1e-5	0.005	1e-5
Init learning rate	0.005	0.01	5e-4	0.005

To assign the credit of each key component in our DeepGraph model, we also experiment with some of its variants, by feeding our graph descriptor (**GD**) to a linear regressor (**GD-linear**), a standard convolutional neural network (**GD-CNN**), and a multilayer perceptron (**GD-MLP**). Hyper-parameters for these models are tuned similarly as baselines.

5. EXPERIMENT RESULTS

5.1 Overall performance

Table 3: Performance measured by MSE (the lower the better), where original label y is scaled to $\log_2(y + 1)$.

Dataset	Facebook	YouTube	AAN	IMDB
Feature-deep	1.107	2.623	0.421	0.527
Feature-linear	1.116	2.633	0.439	0.525
GK-smooth	1.313***	2.675***	0.480***	0.561**
GK-deep	1.315***	2.671***	0.492***	0.565**
GK-linear	1.335***	2.736***	0.519**	0.576***
WL-smooth	1.158***	2.659	0.434	0.536
WL-deep	1.165**	2.654	0.437	0.532
WL-linear	1.331***	2.702***	0.445	0.596***
SP-smooth	1.138	2.615	0.422	0.530
SP-deep	1.155**	2.607	0.428	0.531
SP-linear	1.179***	2.613	0.432	0.535
PSCN	1.117	2.534***	0.425	0.528
Proposed methods				
GD-linear	1.174***	2.750***	0.587***	0.583***
GD-MLP	1.082*	2.427***	0.394***	0.513*
GD-CNN	1.087	2.429***	0.391***	0.512*
DeepGraph	$\mathbf{1.068}^{**}_{\triangledown}$	$\mathbf{2.409}^{***}_{\triangledown\triangledown}$	**0.379*****	$\mathbf{0.508}^{***}_{\triangledown}$

"***(**)" means the result is significantly better or worse over *Features-dp* according to paired t-test test at level 0.01(0.1). "▽" means DeepGraph-multi is better than the better one between DeepGraph-MLP and DeepGraph-CNN.

The overall performance of all competing methods across data sets are displayed in 3. We make the following observations. First, integrating graph descriptor with deep learning, our method DeepGraph outperforms all competing methods significantly. This empirically confirms that graph descriptor could preserve more information of the network structure than bag-of-substructures, both globally and locally. In contrast, utilizing manually designed features could lead to loss of information.

GD-MLP and GD-CNN have already gain improvement over the strongest baseline on most of the data sets, while DeepGraph can further improve the performance by utilizing the semantics of HKS-based graph descriptor. This shows that we can indeed extract more useful features by applying column-wise and row-wise convolution over graph descriptors.

Comparing with GD-linear, which applies linear regression on top of the HKS-based graph descriptor, DeepGraph, GD-MLP, and GD-CNN performs significantly better. This indicates that the effectiveness of the HKS-based graph descriptor has to be utlized by a "deeper" model which explores the convolutions and non-linear transformations of the low-level representation.

Comparing feature based methods with other baselines, the former exhibit strong prediction power. Incorporating both local and global information, the hand-crafted features are very indicative of network growth, which is hard for automatic methods to compete.

When trained on deep networks, the performance of graph kernels could be improved over their linear version. Smoothing kernels can further bring in some improvement. By applying convolution over locally connected regions of the graphs, PSCN can beat many graph kernels on most data sets. These results are consistent with previous studies [24, 41].

5.2 Computational Cost of DeepGraph

Training of DeepGraph is very fast. The models are converged in less than 10 minutes on a Titan X GPU. The major overhead of DeepGraph is the computation of the HKS-based graph descriptors. We empirically measure the computation time for all data sets on a server with 2.40 GHz CPU and 120G RAM. The graphs in our data sets have size as large as 5,000 nodes and 200,000 edges, which is enough for most network prediction problems [20, 27, 42]. The generation of graph descriptors takes an average of 0.86 hour per data set. In contrast, the strongest baseline, feature based method, takes 7.9 hours on average to generate all features. While the strongest graph kernel, SP, takes nearly 5 days.

5.3 Feature Analysis

It has been shown empirically that DeepGraph could well abstract high-level features to represent graphs. It is intriguing to know whether these learned features correspond to well-known structural patterns in network literature. To this end, we select some of the network properties manually computed for the feature based method. Note that we work only on test set, as we care more about the prediction performance. These properties characterize either global or local aspects of networks, and are listed in Figure 3.

The feature vectors output by the last hidden layer of DeepGraph are fed to t-SNE [4], a dimensionality reduction algorithm for visualizing high-dimensional data sets. The visualizations of data set AAN are displayed in Figure 3. We obtain similar results on other data sets, which are omitted to conserve space.

To connect the hand-crafted structural properties with the learned high-level features, we color individual graphs by the values of these properties (e.g., network density). Patterns on the distribution of colors could suggest a connection between learned features and the network property.

Some observations can be made from Figure 3. First, as the number of open and closed triangles are actually features of graphlets [29, 36], we can see that DeepGraph has automatically learned these useful features without human input. Second, since edge density is a function of the number of edges and nodes, DeepGraph not only learns the number of edges and nodes (we do not show the edge and node property in Figure 3, but this is true), but also their none-linear relationship that involves division.

5.4 Error Analysis

Graphs in our data sets typically have hundreds of nodes, which is hard for humans to directly generalize useful information from a set of graphs. As a compromise, we characterize graphs by a set of simple network properties, e.g., the number of nodes, edges, and edge density.

We first want to investigate graphs for which DeepGraph makes more mistakes than baseline, and also the other way around. Here we use the strongest baseline, feature-based method as our reference. The procedure is as follows: among graphs where DeepGraph has smaller MSE than the baseline, we select the top 100 with the largest MSE differences between the two methods. For these top graphs, we compute the average of the properties mentioned above. Similar procedure is also applied to the baseline.

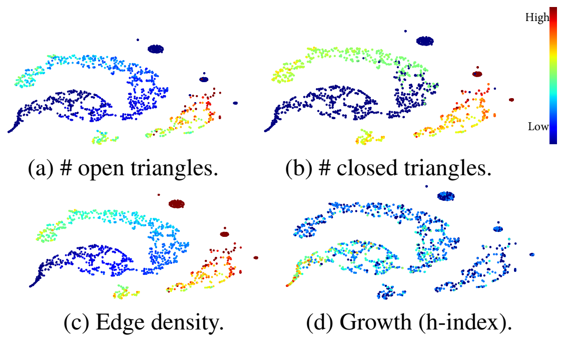

Figure 3: Feature visualization from Data set AAN. One point is a graph in *test* set. The layout is produced from high-level representations of DeepGraph, colored using structural, *hand-crafted* network properties, which are presented under each subfigures. Red (blue) color indicates high (low) property values.

The statistics of graphs where either DeepGraph or the baseline significantly outperforms the other are higher than the average statistics of each data set. This could result form the skewed distribution of the data set – a large number of graphs are of smaller size, leading to more training instances of small graphs. We also observe that both methods perform reasonably well on denser networks.

On the other hand, graphs on which DeepGraph performed better have relatively larger sizes than those where the baseline performed better. This indicates that the HKS representation has an advantage on larger graphs, the structures of which are more difficult to be represented by a bag of local substructures.

6. CONCLUSION

We present a novel neural network model that predicts the growth of egonet properties based on its graph structure. This model, DeepGraph, computes a new representation of the graph structure based on heat kernel signatures. A multi-column, multi-resolution convolution neural network is designed to further learn the high-level representations and predict the network growth in an end-to-end fashion. Experiments on large collections of real-world networks prove that DeepGraph significantly outperforms methods based on hand-crafted features, graph kernels, and competing deep learning methods. The higher-level representations learned by DeepGraph well correlate with findings and theories in social network literature, showing that a deep learning model can automatically discover meaningful and predictive structural patterns in networks.

Our study reassures the predictive power of network structures and suggests a way to effectively utilize this power. A meaningful future direction is to integrate network structure with other types of information, such as the content of information cascades in the network. A joint representation of multi-modal information may maximize the performance of particular prediction tasks.

7. REFERENCES

[1] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan. Group formation in large social networks: membership, growth, and evolution. In *Proc. of SIGKDD*, 2006.

[2] L. Bai, L. Rossi, A. Torsello, and E. R. Hancock. A quantum jensen–shannon graph kernel for unattributed graphs. *Pattern Recognition*, 2015.

[3] X. Bai and E. R. Hancock. Heat kernels, manifolds and graph embedding. In *Structural, Syntactic, and Statistical Pattern Recognition*. Springer, 2004.

[4] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *JSTAT*, 2008.

[5] K. M. Borgwardt and H.-P. Kriegel. Shortest-path kernels on graphs. In *Proc. of ICDM*, 2005.

[6] R. S. Burt. The network structure of social capital. *Research in organizational behavior*, 2000.

[7] R. Chen, Y. Chen, Y. Liu, and Q. Mei. Does team competition increase pro-social lending? evidence from online microfinance. *Games and Economic Behavior*, 2015.

[8] J. Cheng, L. Adamic, P. A. Dow, J. M. Kleinberg, and J. Leskovec. Can cascades be predicted? In *Proc. of WWW*, 2014.

[9] F. Chung. Laplacians and the cheeger inequality for directed graphs. *Annals of Combinatorics*, 2005.

[10] B. G. P. M. Dragomir R. Radev, Mark Thomas Joseph. A Bibliometric and Network Analysis of the field of Computational Linguistics. *JASIST*, 2009.

[11] D. Easley and J. Kleinberg. *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge University Press, 2010.

[12] Y. Fang, M. Sun, and K. Ramani. Heat-passing framework for robust interpretation of data in networks. *PloS one*, 2015.

[13] Y. Fang, J. Xie, G. Dai, M. Wang, F. Zhu, T. Xu, and E. Wong. 3d deep shape descriptor. In *Proc. of CVPR*, 2015.

[14] T. Gärtner, P. Flach, and S. Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Learning Theory and Kernel Machines*. 2003.

[15] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proc. of SIGKDD*, 2016.

[16] R. Guimera, B. Uzzi, J. Spiro, and L. A. N. Amaral. Team assembly mechanisms determine collaboration network structure and team performance. *Science*, 2005.

[17] H. Kashima, K. Tsuda, and A. Inokuchi. Kernels for graphs. *Kernel methods in computational biology*, 2004.

[18] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *Proc. of ICLR*, 2015.

[19] G. Kossinets and D. J. Watts. Empirical analysis of an evolving social network. *science*, 2006.

[20] A. Kupavskii, L. Ostroumova, A. Umnov, S. Usachev, P. Serdyukov, G. Gusev, and A. Kustarev. Prediction of retweet cascade size over time. In *Proc. of CIKM*, 2012.

[21] H. Ma, H. Yang, M. R. Lyu, and I. King. Mining social networks using heat diffusion processes for marketing candidates selection. In *Proc. of CIKM*, 2008.

[22] A. Mislove. *Online Social Networks: Measurement, Analysis, and Applications to Distributed Information Systems*. PhD thesis, 2009.

[23] A. Narayanan, M. Chandramohan, L. Chen, Y. Liu, and S. Saminathan. subgraph2vec: Learning distributed representations of rooted sub-graphs from large graphs. In *Workshop on Mining and Learning with Graphs*, 2016.

[24] M. Niepert, M. Ahmed, and K. Kutzkov. Learning convolutional neural networks for graphs. 2016.

[25] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proc. of SIGKDD*, 2014.

[26] S. Ranu and A. K. Singh. Graphsig: A scalable approach to mining significant subgraphs in large graph databases. In *Proc. of ICDE*, 2009.

[27] D. M. Romero, C. Tan, and J. Ugander. On the interplay between social and topical structure. *Proc. of ICWSM*, 2013.

[28] H. Saigo, S. Nowozin, T. Kadowaki, T. Kudo, and K. Tsuda. gboost: a mathematical programming approach to graph classification and regression. *Machine Learning*, 2009.

[29] N. Shervashidze, T. Petri, K. Mehlhorn, K. M. Borgwardt, and S. Vishwanathan. Efficient graphlet kernels for large graph comparison. In *International conference on artificial intelligence and statistics*, pages 488–495, 2009.

[30] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt. Weisfeiler-lehman graph kernels. *JMLR*, 2011.

[31] J. Sun, M. Ovsjanikov, and L. Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *Computer graphics forum*, 2009.

[32] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. In *Proc. of VLDB*, 2011.

[33] G. Szabo and B. A. Huberman. Predicting the popularity of online content. *Communications of the ACM*, 2010.

[34] J. Tang, M. Qu, and Q. Mei. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proc. of SIGKDD*, 2015.

[35] O. Tsur and A. Rappoport. What's in a hashtag?: content based prediction of the spread of ideas in microblogging communities. In *Proc. of WSDM*, 2012.

[36] J. Ugander, L. Backstrom, and J. Kleinberg. Subgraph frequencies: Mapping the empirical and extremal geography of large graph collections. In *Proc. of WWW*, 2013.

[37] J. Ugander, L. Backstrom, C. Marlow, and J. Kleinberg. Structural diversity in social contagion. *Proc. of the National Academy of Sciences*, 2012.

[38] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi. On the evolution of user interaction in facebook. In *Proc. of WOSN*, 2009.

[39] J. Xie, Y. Fang, F. Zhu, and E. Wong. Deepshape: Deep learned shape descriptor for 3d shape matching and retrieval. In *Proc. of CVPR*, 2015.

[40] P. Yanardag and S. Vishwanathan. Deep graph kernels. In *Proc. of SIGKDD*, 2015.

[41] P. Yanardag and S. Vishwanathan. A structural smoothing framework for robust graph comparison. In *NIPS*, 2015.

[42] J. Yang and J. Leskovec. Modeling information diffusion in implicit networks. In *Proc. of ICDM*, 2010.

[43] L. Yang, T. Sun, M. Zhang, and Q. Mei. We know what@ you# tag: does the dual role affect hashtag adoption? In *Proc. of WWW*, 2012.