Node Classification in Temporal Graphs through Stochastic Sparsification and Temporal Structural Convolution

Cheng Zheng[1] (✉), Bo Zong[2], Wei Cheng[2], Dongjin Song[2], Jingchao Ni[2], Wenchao Yu[2], Haifeng Chen[2], and Wei Wang[1]

[1] Department of Computer Science, University of California, Los Angeles, CA, USA
{chengzheng, weiwang}@cs.ucla.edu
[2] NEC Laboratories America, Princeton, NJ, USA
{bzong, weicheng, dsong, jni, wyu, haifeng}@nec-labs.com

Abstract. Node classification in temporal graphs aims to predict node labels based on historical observations. In real-world applications, temporal graphs are complex with both graph topology and node attributes evolving rapidly, which poses a high overfitting risk to existing graph learning approaches. In this paper, we propose a novel Temporal Structural Network (TSNet) model, which jointly learns temporal and structural features for node classification from the sparsified temporal graphs. We show that the proposed TSNet learns how to sparsify temporal graphs to favor the subsequent classification tasks and prevent overfitting from complex neighborhood structures. The effective local features are then extracted by simultaneous convolutions in temporal and spatial domains. Using the standard stochastic gradient descent and backpropagation techniques, TSNet iteratively optimizes sparsification and node representations for subsequent classification tasks. Experimental study on public benchmark datasets demonstrates the competitive performance of the proposed model in node classification. Besides, TSNet has the potential to help domain experts to interpret and visualize the learned models.

Keywords: Temporal graphs · node classification · graph sparsification · temporal structural convolution

1 Introduction

Temporal graphs, as a data structure that carries both temporal and structural information from real-world data, has been widely adopted in applications from various domains, such as online social media [33], biology [27], action recognition [28], and so on. In this paper, we study the problem of node classification in temporal graphs [27]: Given a set of nodes with rich features and a temporal graph that records historical activities between nodes, the goal is to predict the label of every node. Consider the following application scenario.

Example. In the financial domain, investors are eager to know which companies are promising for investment. As shown in Figure 1, companies and their

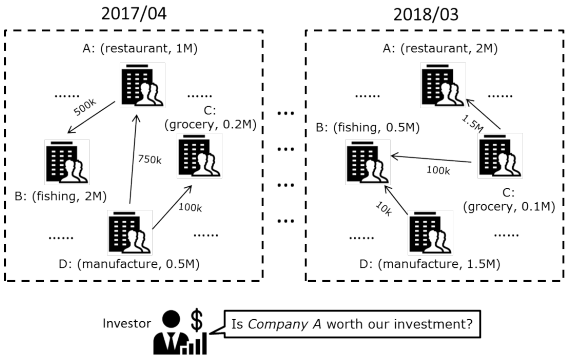


Fig. 1. An example of node classification in a temporal graph from the financial domain. Nodes are companies, and edges indicate monthly transactions. The goal is to predict which companies are promising for investment in the near future.

historical transactions naturally form a temporal graph, shown as a sequence of graph snapshots. Each snapshot encodes companies as nodes and transactions as edges within a month. The side information of companies (*e.g.,* industry sector and cash reserve) and transactions (*e.g.,* transaction amount) is represented by the node and edge attributes, respectively. In this task, we aim to predict each company's label: *promising* or *others* for future investment, with interpretable evidence for domain experts.

While node representation lies at the core of this problem, we face two main challenges from temporal graphs.

Temporal graph sparsification. Temporal graphs from real-life applications are large with high complexity. For example, the social graph on Facebook [5] and the financial transaction graph on Venmo [32] are densely connected with average node degrees of 500 and 111, respectively. Such complexity poses a high overfitting risk to existing machine learning techniques [20, 17], and makes it difficult for domain experts to interpret and visualize learned models. While graph sparsification [16] suggests a promising direction to reduce graph complexity, existing methods perform sparsification by sampling subgraphs from predefined distributions [14, 11, 4, 30]. The sparsified graphs may miss important information for classification because the predefined distributions could be irrelevant to subsequent tasks. Several recent efforts [8, 22, 34] strive to utilize supervision signals to remove noise edges and regularize the graph model training. However, the proposed methods are either transductive with difficulty to scale or of high gradient variance bringing increased training difficulty.

Temporal-structural convolution. Local features in the temporal-structural domain are the key to node classification in temporal graphs. Although existing techniques have investigated how to build convolutional operators to automatically learn and extract local features from either temporal domain [2] or structural domain [25], a naïve method that simply stacks temporal and structural

operators could lead to suboptimal performance. An effective method that learns and extracts local features from joint temporal-structural space is still missing.

Our contribution. We propose Temporal Structural Network (TSNet), a deep learning framework that performs supervised node classification in sparsified temporal graphs. TSNet consists of two major sub-networks: *sparsification network* and *temporal-structural convolutional network*.

1. The *sparsification network* aims to sparsify input temporal graphs by sampling edges from the one-hop neighborhood following a distribution that is learned from the subsequent supervised classification tasks.
2. The *temporal-structural convolutional network* takes sparsified temporal graphs as input and extracts local features by performing convolution in nodes' neighborhood defined in joint temporal-structural space.

As both sub-networks are differentiable, we can leverage standard stochastic gradient descent and backpropagation techniques to iteratively learn better parameters to sparsify temporal graphs and extract node representations. Experimental results on both public and private datasets show that TSNet can offer competitive performance on node classification tasks. Using a case study, we demonstrate the potential of TSNet to improve model interpretation and visualization of temporal graphs.

2 Problem Definition

In the following presentation, we use bold uppercase letters (*e.g.,* $\mathbf{A}$) to represent tensors, uppercase letters (*e.g.,* A) to represent matrices, lowercase letters (*e.g.,* a) to denote scalars, and blackboard bold uppercase letters (*e.g.,* $\mathbb{A}$) to denote the concept of sets. We start with the definition of temporal graphs.

Temporal graphs. In this work, we employ graph snapshot sequences to represent temporal graphs. Given t discrete time points and n nodes, a temporal graph is denoted as $G = (\mathbb{V}, \mathbf{V}, \mathbf{E}, \mathbf{A})$, where $\mathbb{V}$ is a set of nodes that appear in a temporal graph and $\mathbf{V} \in \mathbb{R}^{t \times n \times d_n}$ is a tensor that encodes d_n-dimensional node attributes across t time points. $\mathbf{E} \in \mathbb{R}^{t \times n \times n}$ is a binary tensor where $\mathbf{E}(i, u, v) = 1$ if there is an edge between node u and node v at time i. $\mathbf{A} \in \mathbb{R}^{t \times n \times n \times d_e}$ is a tensor that encodes d_e-dimensional edge attributes across t time points.

Node classification in temporal graphs. Given a temporal graph $G = (\mathbb{V}, \mathbf{V}, \mathbf{E}, \mathbf{A})$ representing historical transactions and $\mathbb{Y}$ as a set of possible node labels, the goal is to predict labels $Y(u)$ for each node $u \in \mathbb{V}$.

In the following discussion, we focus on the cases where node labels are static for the ease of presentation. Note that with minor modification, our technique can easily be adapted for the cases where node labels dynamically evolve. We approach this problem by inductive supervised learning. In the training phase, we are given a temporal graph G_{train} with training labels Y_{train}. In the testing phase, we use the trained model to infer node labels in testing graph G_{test}.

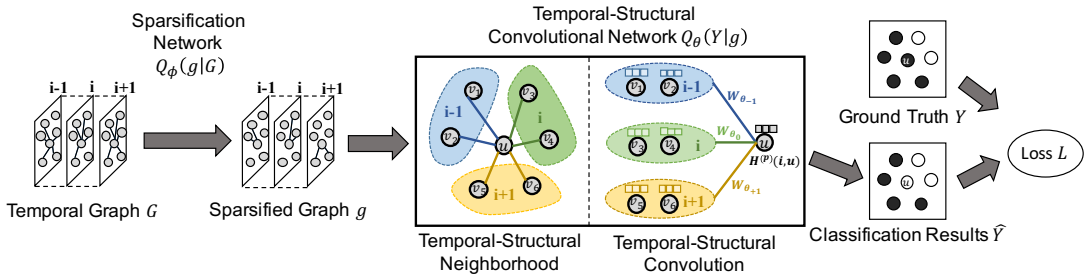


Fig. 2. The frameworks of TSNet. We utilize the two-step formulation of node classification problem. The sparsification network takes the temporal graph as input and generates sparsified subgraphs drawn from a learned distribution. The temporal-structural network extracts temporal and structural features simultaneously with the sparsified subgraph as input.

3 TSNet Overview

In this section, we start with a theoretical overview of the proposed TSNet.

3.1 A Two-step Framework

Given input temporal graph G and node label matrix Y, our objective is to learn $P(Y \mid G)$. Current Graph Neural Networks (GNNs) [13, 11, 25] learn node representation by aggregating node neighborhood features. However, in large and complex temporal graphs, node neighborhood tends to be dense with much noise which introduces high overfitting risk to existing approaches. To tackle the challenge, we leverage the two-step framework proposed in [34] to break node classification problem down into two steps: graph sparsification step and representation learning step.

$$P(Y \mid G) \approx \sum_{g \in \mathbb{S}_G} P(Y \mid g) P(g \mid G) \approx \sum_{g \in \mathbb{S}_G} Q_\theta(Y \mid g) Q_\phi(g \mid G) \tag{1}$$

where g is a sparsified subgraph, and $\mathbb{S}_G$ is a class of sparsified subgraphs of G. We approximate the distributions by tractable functions Q_θ and Q_ϕ. With reparameterization tricks [10], we could differentiate the graph sparsification step to make efficient backpropagation. In the following, we will introduce our framework to find approximation function $Q_\phi(g \mid G)$ and $Q_\theta(Y \mid g)$.

3.2 Architecture

As shown in Figure 2, the proposed TSNet consists of two major sub-networks: **sparsification network** and **temporal-structural convolutional network**.

- The **sparsification network** is a multi-layer neural network that implements $Q_\phi(g \mid G)$: Taking temporal graph G as input, it generates a random sparsified subgraph of G drawn from a learned distribution.
- The **temporal-structural convolutional network** implements $Q_\theta(Y \mid g)$ that takes a sparsified subgraph as input, extracts node representations by convolutional filtering on the temporal-structural neighborhood of each node, and makes predictions on node labels.

With differentiable operations in both sub-networks, our TSNet is an end-to-end supervised framework, which is trainable using gradient-based optimization.

4 Sparsification network

In this section, we present the sparsification network, which optimizes temporal graph sparsification for subsequent node classification tasks.

4.1 Design Goals

The goal of sparsification network is to generate sparsified subgraphs for temporal graphs, serving as the approximation function $Q_\phi(g \mid G)$. Therefore, we need to answer the following three questions in the sparsification network.

1. As the essence of sparsification is to sample a subset of edges, how should we represent each edge so that we can differentiate edges for edge sampling?
2. What is the class of sparsified subgraphs $\mathbb{S}_G$? How to sample such sparsified subgraphs?
3. How to make sparsified subgraphs differentiable for end-to-end training?

4.2 Edge Representations

Given a temporal graph $G = (\mathbb{V}, \mathbf{V}, \mathbf{E}, \mathbf{A})$, an expected edge representation could consist of its edge attributes and certain information from the two connected nodes. Let $\mathbb{N}_{u,i}$ be the set of one-hop neighbors with respect to node u's incoming edges at time i. The expected edge representation $\mathbf{X}(i, u, v)$ for the edge from v to u at time i is calculated as follows.

$$\mathbf{X}(i, u, v) = \mathbf{V}'(i, u)||\mathbf{V}'(i, v)||\mathbf{A}(i, u, v) \tag{2}$$

where $||$ indicates vector concatenation and $\mathbf{A}(i, u, v)$ denotes edge attributes. $\mathbf{V}'(i, u)$ ($\mathbf{V}'(i, v)$) is the representation of node u (v), which we calculate with mean aggregation [11] to capture both attribute and structural information,

$$\mathbf{V}'(i, u) = \sigma(W_{\phi_1} \cdot \mathbf{V}(i, u)||\text{MEAN}(\mathbf{V}(i, u'), \forall u' \in \mathbb{N}_{u,i})) \tag{3}$$

where W_{ϕ_1} is the weights to be learned and σ is a nonlinear activation function.

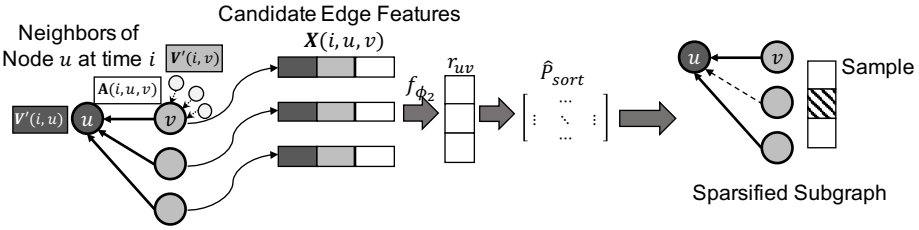


Fig. 3. An illustration of the proposed sparsification network. In this example, we focus on the node u at time i with 3 neighbor nodes and set k as 2. Edge representations consist of both edge attributes and node representations. We implement the sparsification by a continuous relaxation of sorting and top-k important incoming edge sampling.

4.3 Sampling Sparsified Subgraphs

We focus on k-neighbor subgraphs for $\mathbb{S}_G$. The concept of k-neighbor subgraph is originally proposed in the context of spectral sparsification for static graphs [23]: Given an input graph, each node of a k-neighbor subgraph can select no more than k edges from its one-hop neighborhood. In this work, we extend the concept of k-neighbor subgraph to temporal graphs: Given a temporal graph G, each node of a k-neighbor subgraph can select no more than k incoming edges *from its one-hop neighborhood in each graph snapshot of* G. Without loss of generality, we sketch this sampling process by focusing on a specific node u in graph snapshot at time i. Let $\mathbb{N}_{u,i}$ be the set of one-hop neighbors with respect to u's incoming edges at time i and the cardinality of $\mathbb{N}_{u,i}$ is d.

1. For $v \in \mathbb{N}_{u,i}$, $r_{uv} = f_{\phi_2}(\mathbf{X}(i,u,v))$, where r_{uv} is a scalar denoting the ranking score of the edge from node v to u at time i, and f_{ϕ_2} is a feedforward neural network (parameterized by ϕ_2) that generates the score based on the edge representation $\mathbf{X}(i,u,v)$.
2. We sort the incoming edges based on their ranking scores, and select the top-k edges with the largest ranking scores.
3. The above two steps are repeated for each node in each graph snapshot.

The parameters in f_{ϕ_2} are shared among all nodes in all graph snapshots; therefore, the number of parameters is independent to the size of temporal graphs.

4.4 Making Samples Differentiable

The conventional sorting operators are not differentiable such that it is difficult to utilize them for parameter optimization. To make sorting differentiable, we propose to implement the subgraph sampling based on the continuous relaxation of sorting operator [10]. Without loss of generality, we focus on a specific node u at time i in a temporal graph G. We implement the subgraph sampling in Section 4.3 as follows.

1. Let $\mathbb{N}_{u,i}$ be the set of one-hop neighbors with respect to u's incoming edges at time i. We apply the reparameterization trick and introduce a fixed source of randomness [10] to the ranking score r_{uv}, $\forall v \in \mathbb{N}_{u,i}$,
$$\pi_{uv} = \log r_{uv} + g_{uv} \tag{4}$$
where π_{uv} is a reparameterized scalar indicating the importance of the edge from node v to u. g_{uv} is a sample drawn from Gumbel$(0, 1)$ and $g_{uv} = -\log(-\log(u))$ with $u \sim$ Uniform$(0, 1)$. The reparameterization trick refines the stochastic computational graph for smooth gradient backward pass.
2. We relax the permutation matrix of the edge sorting operator $\hat{P}_{sort} \in \mathbb{R}^{d\times d}$ for node u at time i, and its j-th row is
$$\hat{P}_{sort}(j,:)(\tau) = \text{softmax}[((d+1-2j)\pi_{u:} - A_\pi \mathbf{1})/\tau] \tag{5}$$
where d is the cardinality of $\mathbb{N}_{u,i}$ and $\mathbf{1}$ denotes the column vector of all ones. A_π is the matrix of absolute pairwise differences of the elements in $\{v \in \mathbb{N}_{u,i} \mid \pi_{uv}\}$, and the element at x-row and y-column is $A_\pi(x, y) = |\pi_{ux} - \pi_{uy}|$. τ is a hyper-parameter called *temperature* which controls the interpolation between discrete distribution and continuous categorical densities.
3. Before sparsification, the feature tensor of $\mathbb{N}_{u,i}$ is $\mathbf{V}_U(i, u) = \{\mathbf{V}(i, u, u'_1), \ldots, \mathbf{V}(i, u, u'_d)\}$, where $\mathbf{E}(i, u, u'_j) = 1$ and $\mathbf{V}_U(i, u) \in \mathbb{R}^{d\times d_n}$. By applying the relaxed sort operator $\hat{P}_{sort}$ to the unsparsified node features $\mathbf{V}_U(i, u)$, we then select first k rows as the output
$$\mathbf{V}_S(i, u) = [\hat{P}_{sort}\mathbf{V}_U(i, u)](: k, :) \tag{6}$$
If $|\mathbb{N}_{u,i}| \leq k$ for node u, we will skip its sparsification and take all in $\mathbf{V}_U(i, u)$.

Algorithm 1 Sampling subgraphs by sparsification network

Input: Temporal graph $G = (\mathbb{V}, \mathbf{V}, \mathbf{E}, \mathbf{A})$ and integer k.

```
for i = 1, ··· , t do
    for u ∈ V do
        if |N_{u,i}| > k then
            for v ∈ N_{u,i} do
                compute X(i, u, v) by Equation (2)
                compute π_{uv} by Equation (4)
            end for
            compute P̂_sort by Equation (5)
            compute V_S(i, u) by Equation (6)
        end if
    end for
end for
```

We sketch the full algorithm of sparsification network in a combinatorial manner in Algorithm 1. Let $\bar{d}$ be the average degree, n be the total number of

nodes in a temporal graph, and t be the number of snapshots. The sparsification network visits each node's one-hop neighborhood and makes $\bar{d}^2$ calculations. The complexity of sampling subgraphs by the sparsification network is $O(\bar{d}^2 nt)$.

5 Temporal-Structural Convolutional Network

As discussed in Section 3.1, the goal of the temporal-structural convolutional network (TSCN) is to serve as $Q_\theta(Y \mid g)$: it extracts node representations from the sparsified subgraphs generated by the sparsification network and leverages the vector representations to perform node classification. Inspired by the success of convolutional aggregation in the graph domain [11, 13, 4, 25], the core idea behind the temporal-structural convolutional network is to simultaneously extract local temporal and structural features for node representations by convolutional aggregation in individual nodes' *temporal-structural neighborhood*.

5.1 Temporal-structural Neighborhood

Unlike the neighborhood defined in static graphs that only tells "who are close to me", temporal-structural neighborhood stores information about "who and when are close to me". To accomplish this, we extend the notion of neighborhood to the temporal domain by aggregating the structural neighborhood across several preceding and/or subsequent snapshots of any given snapshot. Given a node u at time i, its temporal-structural neighborhood can be represented by a matrix $F_{u,i} \in \mathbb{R}^{t \times n}$, where $F_{u,i}(j, v) = 1$ if node v is in u's (structural) neighborhood at time j; otherwise, $F_{u,i}(j, v) = 0$. In this work, we focus on the *first-order* temporal-structural neighborhood in the sparsified subgraphs. In other words, we have $F_{u,i}(j, v) = 1$ if the following two conditions hold: (1) $|i - j| = 1$, and (2) at time j, there is an incoming edge from node v to u in the sparsified temporal graph. Note that node u at time i is also in its own temporal-structural neighborhood, that is, $F_{u,i}(i, u) = 1$. With the notion of the temporal-structural neighborhood, we are ready to introduce the design of a temporal-structural convolutional layer.

5.2 Temporal-structural Convolutional Layer

A temporal-structural convolutional layer performs feature aggregation in individual nodes' temporal-structural neighborhood. One could stack multiple convolutional layers to extract higher-order temporal-structural features.

Without loss of generality, we discuss the technical details of temporal-structural convolutional layer by focusing on a specific node u at time i in the p-th convolutional layer. The input is a temporal graph $G = (\mathbb{V}, \mathbf{V}, \mathbf{E}, \mathbf{A})$, node representations $\mathbf{H}^{(p-1)} \in \mathbb{R}^{t \times n \times d_n^{(p-1)}}$ and a relaxed sort operator $\hat{P}_{sort}$ from Section 4.4. With the same sort operator in Equation 6 that sparsifies the node

features of the first convolution layer, we obtain the sparsified node features of the p-th convolutional layer as

$$\mathbf{V}_U^{(p)}(i,u) = \{\mathbf{H}^{(p-1)}(i,u,u'_1), \ldots, \mathbf{H}^{(p-1)}(i,u,u'_d)\} \tag{7}$$

$$\mathbf{V}_S^{(p)}(i,u) = [\hat{P}_{sort}\mathbf{V}_U^{(p)}(i,u)](:k,:) \tag{8}$$

The temporal-structural convolution performs as follows.

$$\mathbf{H}^{(p)}(i,u) = \sigma(\sum_{\{(j,v)|F_{u,i}(j,v)=1\}} \mathbf{V}_S^{(p)}(j,u,v)W_{i,u,j,v}^{(p)}) \tag{9}$$

where $\sigma(\cdot)$ is a non-linear activation function, $\mathbf{H}^{(p)} \in \mathbb{R}^{t\times n\times d_n^{(p)}}$ is the output node representations, and $W_{i,u,j,v}^{(p)} \in \mathbb{R}^{d_n^{(p-1)}\times d_n^{(p)}}$ is a customized convolution filter generated by

$$W_{i,u,j,v}^{(p)} = \text{MLP}_{\theta_{i-j}^{(p)}}(\mathbf{V}_S^{(p)}(i,u), \mathbf{V}_S^{(p)}(j,v), \mathbf{A}(j,u,v)) \tag{10}$$

where $\text{MLP}_{\theta_{i-j}^{(p)}}(\cdot)$ is a multi-layer neural network with parameters $\theta_{i-j}^{(p)}$ that generates customized convolutional filters based on node and edge features. In other words, in the case of first-order temporal-structural neighborhood, we utilize three networks $\text{MLP}_{\theta_{-1}^{(p)}}$, $\text{MLP}_{\theta_0^{(p)}}$, and $\text{MLP}_{\theta_1^{(p)}}$ to model the temporal impacts from the temporal-structural neighborhood. Note that $\{\theta_{-1}^{(p)}, \theta_0^{(p)}, \theta_1^{(p)}\}$ are the only parameters in this convolutional layer and are shared by all nodes; therefore, the number of parameters in a temporal-structural convolutional layer is independent of the number of nodes, edges, or time points in a temporal graph.

As described in Equation 9 and 10, for a single node, the computational cost is determined by the MLP structure as a fixed number (c). Therefore, the complexity of convolution layer is $O(cdtn)$ and is generally proportional to the number of nodes (n).

5.3 Network Architecture

Now we present the full temporal-structural convolutional network.

- **Convolutional layer.** As discussed in Section 5.2, one could stack multiple convolutional layers to hierarchically explore high-order temporal-structural neighborhood.
- **Pooling layer.** Let $\mathbf{H}^{(p)} \in \mathbb{R}^{t\times n\times d_n^{(p)}}$ be the output node representations from the p-th convolutional layers. The pooling layer performs another round of aggregation in temporal domain by $H = \text{Pooling}(\mathbf{H}^{(p)})$, where $H \in \mathbb{R}^{n\times d_n^{(p)}}$. Possible pooling operations include *max*, *average*, and *sum* [11].
- **Output layer**. This layer employs a multi-layer neural network and the final output of TSNet is $\hat{Y} = \text{MLP}_{\theta_o}(H)$, where θ_o denotes the parameters.

– **Objective function**. To handle the estimation variance brought by random sampling, the sparsification network generates l sparsified subgraphs and we optimize the parameters in TSNet by minimizing the average loss from the l samples. In particular, the objective function is formulated as follows.

$$J = \frac{1}{l} \sum_{i=1}^{l} L(Y, \hat{Y}_i) \tag{11}$$

The function L is defined by cross entropy loss.

6 Experimental Study

In this section, we evaluate the performance of TSNet using real-life temporal graph datasets from multiple domains. In particular, we compare TSNet with state-of-the-art techniques in terms of classification accuracy and analyze its sensitivity to the hyper-parameters. Moreover, we provide a case study to demonstrate how sparsified subgraphs generated by TSNet could improve visualization. The supplementary material contains more detailed information.

6.1 Datasets

We employ four temporal graph datasets from different domains, including collaboration networks, online social media, and financial marketing. The dataset statistics are summarized in Table 1.

Table 1. Dataset statistics

	DBLP-3	**DBLP-5**	**Reddit**	**Finance**
# nodes	1,662	5,994	128,858	45,542
# edges	33,808	113,062	29,009,401	661,586
# snapshots / # in training	10/5	10/5	31/16	36/18
time granularity	1 Year	1 Year	1 Day	1 Month
# classes	3	5	10	2

DBLP-3/5. The temporal graphs record co-author relationships in the DBLP computer science bibliography[3] from 2001 to 2010, where nodes represent authors and edges denote co-author relationships. There are 10 graph snapshots and each snapshot stores co-author relationships within one year. To generate the node attributes, we aggregate titles and abstracts of the corresponding author's papers published in that year into one document, represent this document

[3] https://www.aminer.cn/citation

by the bag-of-words model. We aim to classify authors in DBLP-3 and DBLP-5 into three and five research areas respectively. The first 5 snapshots are in G_{train}.

Reddit. Reddit is a large online forum, where users contribute original posts or make comments/upvotes to existing posts. We extract posts and comments in 10 mid-sized subreddits in May 2015. Following the procedure in [11], we build a post-to-post temporal graph, where nodes are posts, and two posts become connected if they are both commented by at least one identical user. For node attributes, we aggregate the post and comment texts into one document, represent it by the bag-of-words model, and reduce the dimensionality to 20 by PCA. On this dataset, our goal is to classify each post into one of the 10 subreddit categories. The first 16 snapshots are in G_{train}.

Finance. This private dataset contains temporal graphs that record transaction history between companies from April 2014 to March 2017. Each node represents a company and each edge indicates a transaction between two companies. Node attributes are side information about the companies such as account balance, cash reserve, etc., which may change from year to year. In this dataset, we aim to classify companies into two categories: *promising* or *unpromising* for investment in the near future. We put the first 18 snapshots in G_{train}.

6.2 Baseline Methods

We implement four categories of baselines: (1) node classification methods for static graphs, including **GCN** [13], **GraphSAGE** [11], **GAT** [25], and **LDS** [8] that simultaneously learns the graph structure and GCN parameters; (2) stacking structural and temporal feature learning models, including **TempCNN-GCN** that extracts temporal features with temporal CNN [2] and then trains GCN model on static graphs, **GCN-TempCNN** in reverse order, and **Deepwalk-LSTM** that extracts structure features by DeepWalk [21] and then leverages LSTM to learn temporal features; (3) temporal graph learning models, including **DynamicTriad** [35], and **STAR** [27]; (4) variants of the proposed TSNet models, including **TSCN** that only uses temporal-structural convolutional network, **SS-TSCN** that samples subgraphs with spectral sparsification [23], and **DE-TSCN** that employs DropEdge [22] as graph sampling method.

6.3 Experimental Settings

TSNet. In our experiments, the hyper-parameter k is searched between 3 and 15 for the optimal performance. For the temporal-structural convolutional network, it starts with two temporal-structural convolutional layers, with an internal single-layer feedforward network to generate convolutional filters. Then the output features pass a max-pooling layer over time and a non-linear layer which produces the logit for label prediction.

Dataset split and accuracy metrics. We prepare the training and testing temporal graphs following the similar setting in [11]. We split the snapshots of temporal graphs into G_{train} and G_{test}: test graphs remain unseen during training. We randomly sample 80% nodes in G_{train} as the training nodes and provide

Table 2. Node classification performance

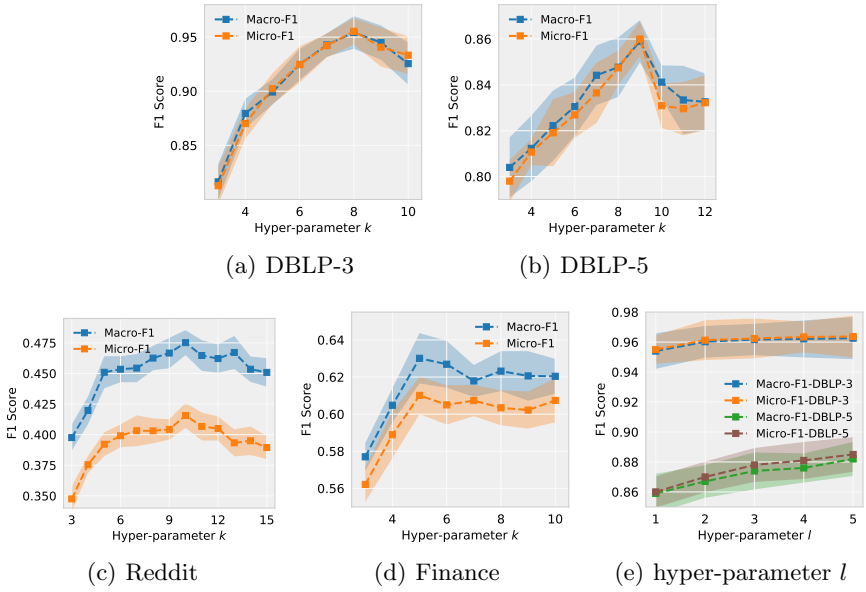
	DBLP-3		**DBLP-5**		**Reddit**		**Finance**	
	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1
GCN	0.862	0.859	0.679	0.678	0.357	0.290	0.480	0.464
GraphSAGE	0.850	0.847	0.814	0.814	0.399	0.336	0.496	0.448
GAT	0.875	0.863	0.821	0.830	-	-	0.509	0.495
LDS	0.876	0.869	0.797	0.794	-	-	0.499	0.474
TempCNN-GCN	0.851	0.857	0.720	0.710	0.411	0.340	0.532	0.548
GCN-TempCNN	0.676	0.691	0.720	0.711	0.384	0.313	0.440	0.397
DeepWalk-LSTM	0.913	0.913	0.772	0.777	0.370	0.303	0.493	0.446
DynamicTriad	0.753	0.745	0.713	0.717	0.393	0.324	0.419	0.430
STAR	0.908	0.908	0.811	0.815	0.439	0.367	0.541	0.502
TSCN	0.942	0.929	0.850	0.845	0.466	0.406	0.559	0.537
SS-TSCN	0.839	0.835	0.807	0.805	0.418	0.351	0.465	0.445
DE-TSCN	0.875	0.888	0.801	0.792	0.391	0.343	0.538	0.487
TSNet	**0.954**	**0.955**	**0.859**	**0.860**	**0.475**	**0.416**	**0.630**	**0.610**

their labels to the models. We test the performance of the models with all nodes in G_{test}. We evaluate the classification accuracy using macro-F1 and micro-F1 scores . The results show the average of 10 runs with random initializations.

6.4 Classification Accuracy

Table 2 summarizes the classification performance of TSNet and the baseline methods on all datasets. For DBLP-3, DBLP-5, Reddit, and Finance, the hyper-parameter k is set as 8, 9, 10, and 5, respectively. The hyper-parameter l is set as 1 in this experiment. Note some results on Reddit is missing due to the out-of-memory error.

Overall, TSNet consistently outperforms all of the baseline methods in terms of macro-F1 and micro-F1 over all of the datasets. We make the following observations. (1) Compared with the deep learning techniques for static graphs, including GCN, GraphSAGE and GAT, TSNet achieves better performance by effectively utilizing temporal features in graphs. (2) In GCN-TempCNN and Deepwalk-LSTM, structural and temporal features are extracted from separate components. Because of the inter-component dependency, it becomes harder to adjust the parameters in structural feature learning than in temporal feature learning. Similarly, in TempCNN-GCN, parameters in temporal feature learning are harder to get trained. With temporal-structural convolution, TSNet can simultaneously adjust parameters for temporal and structural feature learning, and generate more effective features for better classification accuracy. (3) In comparison with LDS, SS-TSCN and DE-TSCN, TSNet outperforms because of the explicit supervision from downstream tasks, which shows that the sparsification network learns to sparsify temporal graphs to favor the subsequent classification. (4) The comparison with TSCN is interesting: using the sparsified



(a) DBLP-3 (b) DBLP-5

(c) Reddit (d) Finance (e) hyper-parameter l

Fig. 4. Accuracy vs hyper-parameter k and l

subgraphs from the sparsification network, it is easier to make TSCN generalized to unseen testing data with improved classification performance. (5) Compared with DynamicTriad and STAR, TSNet is more effective in node representations customized by the simultaneous learning of temporal and structural features.

6.5 Sensitivity to Hyper-parameters

Figure 4(a)-(d) demonstrates how classification accuracy responds when k increases from 3. Over the four datasets, we observe a common phenomenon: there exists an optimal k that delivers the best classification accuracy. When k is small, TSNet can only make use of little structural information, which leads to sub-optimal performance. When k gets larger, the temporal-structural convolution involves more complex neighborhood aggregation with higher overfitting risk, which negatively impacts the classification performance for unseen testing data. By comparing across datasets, we observe that the optimal k is associated with the average node degrees of the temporal graphs: higher k in dense Reddit graph and lower k in sparse Finance graph. Figure 4(e) shows how hyper-parameter l impacts classification accuracy on DBLP-3 and DBLP-5. When l increases from 1 to 5, we observe a relatively small improvement in classification accuracy. As the parameters in the sparsification network are shared by all edges in the temporal graphs, the estimation variance from random sampling could already be mitigated to some extent by a number of sampled edges in a sparsified subgraph.

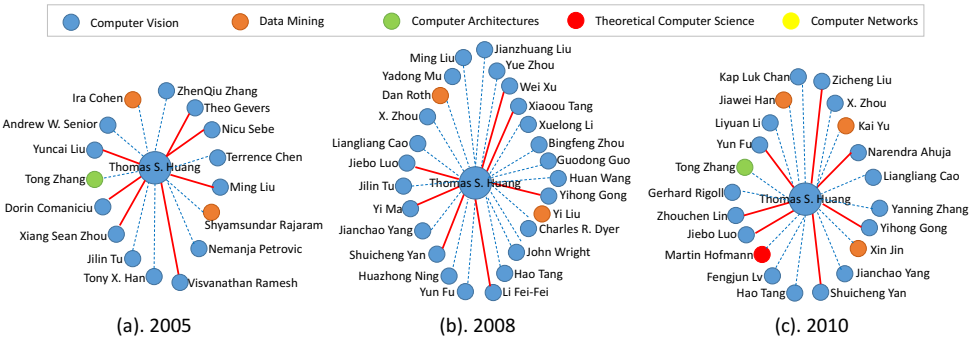


Fig. 5. One-hop neighborhood of *Thomas S. Huang* in DBLP-5. The visualization presents snapshots from (a) 2005, (b) 2008, and (c) 2010. Node colors indicate node labels from ground truth. Sparsification network selects edges with solid red lines. The sparsified graph supports downstream classification as well as model interpretation.

6.6 Case Study

In this section, we present a case study to demonstrate the potential of TSNet in enhancing model interpretation and visualization. Figure 5 visualizes the one-hop neighborhood of *Thomas S. Huang* with k set as 7 in DBLP-5 temporal graph, and we only present snapshots from 2005, 2008, and 2010 due to space limitation. We make the following observations from Figure 5. First, the central author is from the area of *computer vision*, and all selected edges also connect to authors from *computer vision*. When neighbors share identical labels consistently over time, temporal-structural features could boost the confidence of making classification decision. Second, instead of exploring all the neighbors, we can only focus on a subset of selected neighbors, which could make it easier for human experts to conduct the effort on model interpretation and result visualization.

7 Related Work

Our research is related to three lines of studies: graph sparsification, deep learning on graphs, and temporal graph modeling.

Graph sparsification. The goal of graph sparsification is to find small subgraphs from input large graphs that best preserve desired properties. Existing techniques are mainly unsupervised and deal with simple graphs without node/edge attributes for preserving predefined graph metrics [12], information propagation traces [20], graph spectrum [23, 1], node degree distribution [7], node distance distribution [14], or clustering coefficient [19]. Importance based edge sampling has also been studied in a scenario where we could predefine the importance of edges [4]. Unlike existing sparsification methods, our method aims to optimize temporal graph sparsification with rich node and edge attributes by supervision signals from subsequent classification tasks.

Deep graph learning. Deep graph learning has made steady progress on automated node representation learning. Early studies [6, 3] investigate convolutional filters in graph spectral domain under transductive settings. To enable inductive learning, convolutional filters in graph domain are proposed [13, 25]. The graph feature learning models based on message passing mechanisms are also referred to as GNNs. Multiple latest studies [18, 24] extend the GNNs into temporal graphs by introducing the recurrent operations into the GNN layers. Recent efforts also attempt to sample subgraphs from predefined distributions [31, 8], and regularize graph learning by random edge dropping [22]. Our research investigates node representation learning in the temporal structural domain for general temporal graphs, where both node/edge attributes and graph topology could evolve over time. More importantly, our work answers a unique question: Given a limited budget for edge selection, how to optimize graph sparsification so that we could maximize performance in subsequent classification tasks.

Temporal graph modeling. It is an important yet challenging task to model dynamics and evolution in temporal graphs [9, 28, 27]. Recent studies attempt to model dynamics in temporal graphs using generative methods [9], matrix factorization [15], and deep learning approaches [28, 29]. The model in [29] fuses the sequential and spatial graph convolution in a "sandwich" structure, which is yet dependent on the convolution order of sub-structure [26]. The ST-GCN model in [28] learns both spatial and temporal patterns by adding all temporally connected nodes into temporal kernels and conduct similar convolution as GCN [13] on skeleton graph with static topology. Comparing with these approaches, our model TSNet is novel as it can extract the temporal and structural features simultaneously and separate the temporal impact of different convolution filters.

8 Conclusion

In this paper, we propose Temporal Structural Network (TSNet) for node classification in temporal graphs. TSNet consists of two major sub-networks: (1) the sparsification network sparsifies input temporal graphs by sorting and sampling edges following a learned distribution; (2) the temporal-structural convolutional network performs convolution on the sparsified graphs to extract local features from the joint temporal-structural space. As an end-to-end model, the two sub-networks in TSNet are trained jointly and iteratively with supervised loss, gradient descent, and backpropagation techniques. In the experimental study, TSNet demonstrates superior performance over four categories of baseline models on public and private benchmark datasets. The qualitative case study suggests a promising direction for the interpretability of temporal graph learning.

Acknowledgement

We thank the anonymous reviewers for their careful reading and insightful comments on our manuscript. The work was partially supported by NSF (DGE-1829071, IIS-2031187).

References

1. Adhikari, B., Zhang, Y., Amiri, S.E., Bharadwaj, A., Prakash, B.A.: Propagation-based temporal network summarization. TKDE (2018)
2. Bai, S., Kolter, J.Z., Koltun, V.: An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv:1803.01271 (2018)
3. Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and locally connected networks on graphs. In: ICLR (2014)
4. Chen, J., Ma, T., Xiao, C.: Fastgcn: fast learning with graph convolutional networks via importance sampling. In: ICLR (2018)
5. Ching, A., Edunov, S., Kabiljo, M., Logothetis, D., Muthukrishnan, S.: One trillion edges: Graph processing at facebook-scale. VLDB (2015)
6. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: NIPS (2016)
7. Eden, T., Jain, S., Pinar, A., Ron, D., Seshadhri, C.: Provable and practical approximations for the degree distribution using sublinear graph samples. In: WWW (2018)
8. Franceschi, L., Niepert, M., Pontil, M., He, X.: Learning discrete structures for graph neural networks. In: ICML (2019)
9. Ghalebi, E., Mirzasoleiman, B., Grosu, R., Leskovec, J.: Dynamic network model from partial observations. In: NIPS (2018)
10. Grover, A., Wang, E., Zweig, A., Ermon, S.: Stochastic optimization of sorting networks via continuous relaxations. In: ICLR (2019)
11. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: NIPS (2017)
12. Hübler, C., Kriegel, H.P., Borgwardt, K., Ghahramani, Z.: Metropolis algorithms for representative subgraph sampling. In: ICDM (2008)
13. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2017)
14. Leskovec, J., Faloutsos, C.: Sampling from large graphs. In: KDD (2006)
15. Li, J., Cheng, K., Wu, L., Liu, H.: Streaming link prediction on dynamic attributed networks. In: WSDM (2018)
16. Liu, Y., Safavi, T., Dighe, A., Koutra, D.: Graph summarization methods and applications: A survey. ACM Computing Surveys (2018)
17. Loukas, A., Vandergheynst, P.: Spectrally approximating large graphs with smaller graphs. In: ICML (2018)
18. Ma, Y., Guo, Z., Ren, Z., Zhao, E., Tang, J., Yin, D.: Streaming graph neural networks. arXiv:1810.10627v2 (2019)
19. Maiya, A.S., Berger-Wolf, T.Y.: Sampling community structure. In: WWW (2010)
20. Mathioudakis, M., Bonchi, F., Castillo, C., Gionis, A., Ukkonen, A.: Sparsification of influence networks. In: KDD (2011)
21. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: KDD (2014)

22. Rong, Y., Huang, W., Xu, T., Huang, J.: Dropedge: Towards deep graph convolutional networks on node classification. In: ICLR (2020)
23. Sadhanala, V., Wang, Y.X., Tibshirani, R.: Graph sparsification approaches for laplacian smoothing. In: AISTATS (2016)
24. Trivedi, R., Farajtabar, M., Biswal, P., Zha, H.: Dyrep: Learning representations over dynamic graphs. In: ICLR (2019)
25. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: ICLR (2018)
26. Wu, Z., Pan, S., Long, G., Jiang, J., Zhang, C.: Graph wavenet for deep spatial-temporal graph modeling. In: IJCAI (2019)
27. Xu, D., Cheng, W., Luo, D., Liu, X., Zhang, X.: Spatio-temporal attentive rnn for node classification in temporal attributed graphs. In: IJCAI (2019)
28. Yan, S., Xiong, Y., Lin, D.: Spatial temporal graph convolutional networks for skeleton-based action recognition. In: AAAI (2018)
29. Yu, B., Yin, H., Zhu, Z.: Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In: IJCAI (2018)
30. Yu, W., Zheng, C., Cheng, W., Aggarwal, C., Song, D., Zong, B., Chen, H., Wang, W.: Learning deep network representations with adversarially regularized autoencoders. In: KDD (2018)
31. Zeng, H., Zhou, H., Srivastava, A., Kannan, R., Prasanna, V.: Graphsaint: Graph sampling based inductive learning method. In: ICLR (2020)
32. Zhang, X., Tang, S., Zhao, Y., Wang, G., Zheng, H., Zhao, B.Y.: Cold hard e-cash: Friends and vendors in the venmo digital payments system. In: ICWSM (2017)
33. Zheng, C., Zhang, Q., Long, G., Zhang, C., Young, S.D., Wang, W.: Measuring time-sensitive and topic-specific influence in social networks with lstm and self-attention. IEEE Access (2020)
34. Zheng, C., Zong, B., Cheng, W., Song, D., Ni, J., Yu, W., Chen, H., Wang, W.: Robust graph representation learning via neural sparsification. In: ICML (2020)
35. Zhou, L.k., Yang, Y., Ren, X., Wu, F., Zhuang, Y.: Dynamic network embedding by modeling triadic closure process. In: AAAI (2018)