Adaptive Neural Network for Node Classification in Dynamic Networks

Dongkuan Xu[1*], Wei Cheng[2*], Dongsheng Luo[1], Yameng Gu[1], Xiao Liu[1],
Jingchao Ni[2], Bo Zong[2], Haifeng Chen[2], and Xiang Zhang[1*]
[1]The Pennsylvania State University [2]NEC Laboratories America, Inc.
{dux19,dul262,yug56,xxl213,xzz89}@psu.edu, {weicheng,jni,bzong,haifeng}@nec-labs.com

***Abstract*—Given a network with the labels for a subset of nodes, transductive node classification targets to predict the labels for the remaining nodes in the network. This technique has been used in a variety of applications such as voxel functionality detection in brain network and group label prediction in social network. Most existing node classification approaches are performed in static networks. However, many real-world networks are dynamic and evolve over time. The dynamics of both node attributes and network topology jointly determine the node labels. In this paper, we study the problem of classifying the nodes in dynamic networks. The task is challenging for three reasons. First, it is hard to effectively learn the spatial and temporal information simultaneously. Second, the network evolution is complex. The evolving patterns lie in both node attributes and network topology. Third, for different networks or even different nodes in the same network, the node attributes, the neighborhood node representations and the network topology usually affect the node labels differently, it is desirable to assess the relative importance of different factors over evolutionary time scales. To address the challenges, we propose AdaNN, an adaptive neural network for transductive node classification. AdaNN learns node attribute information by aggregating the node and its neighbors, and extracts network topology information with a random walk strategy. The attribute information and topology information are further fed into two connected gated recurrent units to learn the spatio-temporal contextual information. Additionally, a triple attention module is designed to automatically model the different factors that influence the node representations. AdaNN is the first node classification model that is adaptive to different kinds of dynamic networks. Extensive experiments on real datasets demonstrate the effectiveness of AdaNN.**

***Index Terms*—Dynamic Networks, Node Classification, Spatio-Temporal, RNN, Attention Mechanism**

I. Introduction

The rapid growth of information in the past decades has led to a vast increase in the volume of information about individual instances and the connections between them. A large part of this data can be organized in the form of network data where each node represents an instance and the edge indicates the connection, such as brain networks [1] and social networks [2]. When dealing with network data, a subset of nodes may be labeled. Node classification leverages the node proximity manifested to extend the labeling so that each node is assigned a label. For example, in brain networks, a node represents a tidy cube of brain tissue called a voxel and the edge indicates the connectivity between voxels. Node classification in a brain network is to classify the voxels into different categories according to their functionality [3]. Node classification helps us understand the underlying functionality of the unknown nodes and their role when interacting with others. Take the brain network as an example. A brain consists of plenty of voxels, but people usually know the functionality of a part of them. This is because it is with high cost to analyze the functionality of all the voxels and the expressing patterns of some voxels are too complicated to recognize. So people are interested in exploring the unknown voxels based on the voxels with available labels.

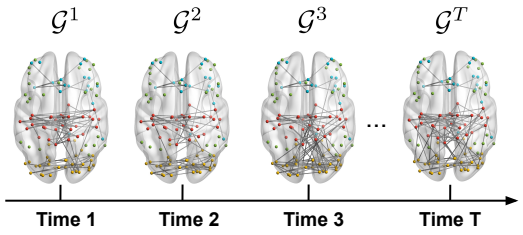


Fig. 1. An illustration of a dynamic brain network.

Due to its importance, there have been considerable research efforts attracted on node classification in recent years [4]–[8]. Perozzi et al. in [4] use the local information extracted by random walks to generate the node representations and predict the node labels with the hierarchical softmax. A convolutional architecture is proposed to classify nodes in networks by considering both node attributes and network topology [6]. Node labels are predicted by the softmax function. The authors in [8] propose an attention-based architecture that performs node classification on network data. These classification approaches are for the static networks.

Many real-world networks are dynamic and network topology evolves over time. Fig. 1 shows the voxel connectivity of the brain network when the subject conducts different tasks at different time steps. The connectivity between voxels will change when the subject conducts different tasks, which results in the edge deletion or addition. This is because the connections between different brain neurons will change under different tasks. Some efforts have been made on the node classification in dynamic networks recently [9]–[11]. Goyal et al. in [10] use autoencoder to capture the nonlinear proximity of nodes and incrementally generate the representations at time t from the ones at time t-1. The triadic closure process is mod-

*These authors contributed equally to this work.

eled to preserve the evolution patterns of dynamic networks in [11]. Nodes are further classified by a logistic regression model. A large part of these approaches are designed for the dynamic networks where network topology evolves over time. However, a majority of networks are with a rich set of node attributes, which also evolve over time. The research on the node classification in the dynamic networks with node attributes is still very limited. Additionally, for different networks, the node attributes and the network topology show differential influence on node representations. Even in the same network, the representations of some nodes can be dominated by node attributes and some are dominated by network topology. These dynamic characteristics motivate us to develop an effective method that can be adaptive to different networks and model different factors influencing the node representations.

In this work, we study the problem of node classification in dynamic networks. Given a sequence of networks for the same set of nodes, where nodes are with attributes and each node has an unique label across difference time steps, the goal is to classify the unlabeled nodes based on the labeled ones. The task is challenging: First, as the spatial and temporal dimensions of dynamic networks are entangled, it is hard to effectively learn node representation based on both spatial and temporal aspects. Second, the dynamic characteristics of dynamic networks are complex. Both node attributes and network topology evolve over time. Third, there are different factors that influence node representations. For example, different neighbors show different influence on node representation. The importance of node representation in different time periods also varies. Another factor is related to the network property. Node representations are influenced by node attributes and network topology differently for different networks.

To address these challenges, we propose AdaNN (Fig. 2.), an adaptive neural network, for node classification in dynamic networks. AdaNN aims to learn node representations for classification by considering the evolution of both node attributes and network topology. More specifically, at each time step, AdaNN learns the node attribute information by aggregating the representation of the node and its neighbors. To extract the network topology information, AdaNN applies a random walk strategy to obtain the structural context of each node. The attribute information and the structural context are further fed into two connected gated recurrent unit (GRU) network [12] to jointly learn the spatio-temporal information of node attributes and network topology. The spatio-temporal information is used as the node representation to classify the node. Moreover, a triple attention module is developed to model different factors that influence the node representations in dynamic networks. In particular, an attention module on spatial aspect helps to differentiate the importance of different neighbors on the target node's representation. On the temporal aspect, an attention module helps to recognize the importance of node representations in different time steps. Another attention module helps to differentiate the relative influence of node attribute and network topology for different networks.

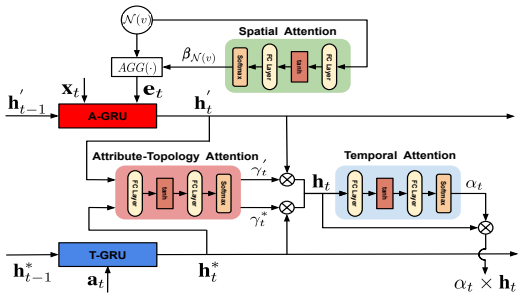


Fig. 2. Overall architecture of AdaNN.

II. The Problem

A dynamic network is a collection of snapshots of an attributed network at different time steps, denoted by $\mathbb{G} = (\mathcal{G}^1, \mathcal{G}^2, \cdots, \mathcal{G}^T)$. $\mathcal{G}^t = (\mathcal{V}, \mathbf{A}^t, \mathbf{X}^t)$ is the network at time step t. The set of nodes $\mathcal{V}$ is fixed for all time steps. Each node has a consistent label across T time steps. $\mathbf{A}^t \in \mathbb{R}^{N\times N}$ is the adjacency matrix and $\mathbf{X}^t \in \mathbb{R}^{N\times d}$ is the node attribute matrix. Both $\mathbf{A}^t$ and $\mathbf{X}^t$ change at different time steps. Given $\mathbb{G}$ and the labels of a subset of nodes $\mathcal{V}_L$, the goal of node classification in dynamic networks is to classify the nodes in subset $\mathcal{V}_U$ whose labels are unknown, where $\mathcal{V} = \mathcal{V}_L \cup \mathcal{V}_U$.

III. Adaptive Neural Network for Node Classification in Dynamic Networks

A. Attribute-Topology GRU Network

To integrate both node attribute information and network topology information, we design a neural network consisting of two connected GRUs, called attribute-topology GRU network (AT-GRUs). The intuition behind AT-GRUs is that two GRUs are used to consider the attribute information and the topology information respectively, and the outputs of the two GRUs are concatenated as a joint state vector.

The GRU that considers the topology information is called T-GRU. T-GRU takes the vector containing the topology information related to the node as input and outputs a state vector. In this paper, we use random walk with restart (RWR) [13] to extract the topology information vector for each node which has been shown effective in exploring network data. Given a network at time step t, $\mathcal{G}^t = (\mathcal{V}, \mathbf{A}^t, \mathbf{X}^t)$, and a starting node v, the k-step RWR vector is defined as $\mathbf{p}^{(k)} = c\mathbf{p}^{(k-1)}[(\mathbf{D}^{-1})\mathbf{A}^t] + (1-c)\mathbf{p}^{(0)}$ where $\mathbf{p}^{(k)} \in \mathbb{R}_+^{1\times N}$. $p_u^{(k)}$ represents the probability of node u after k step transitions from v. $\mathbf{p}^{(0)}$ is the initial vector with $p_v^{(0)}$ = 1 and all other entries equal 0. $\mathbf{D}$ is a diagonal matrix with $\mathbf{D}_{ii} = \sum_{j=1}^{N} \mathbf{A}_{ij}^t$. 1 - c is the probability that the random walker will restart from v. Thus, the topology context vector for node v at time step t is defined as

$$\mathbf{a}_t = \sum_{k=1}^{K} \mathbf{p}^{(k)} \tag{1}$$

where K indicates the number of considered steps. We apply the process to all nodes.

T-GRU is described in Eqs. (2)-(5). It takes the topology vectors of a node at different time steps, i.e., $\mathbf{a}_1$, ..., $\mathbf{a}_T \in \mathbb{R}^d$,

as inputs. A state vector $\mathbf{h}_t^* \in \mathbb{R}^{d_h}$ is learned for each time step by applying the following equations iteratively.

$$\mathbf{z}_t^* = \sigma(\mathbf{W}_z^*[\mathbf{a}_t \oplus \mathbf{h}_{t-1}^*] + \mathbf{b}_z^*), \quad (2)$$

$$\mathbf{r}_t^* = \sigma(\mathbf{W}_r^*[\mathbf{a}_t \oplus \mathbf{h}_{t-1}^*] + \mathbf{b}_r^*), \quad (3)$$

$$\tilde{\mathbf{h}}_t^* = \tanh(\mathbf{W}_h^*[\mathbf{a}_t \oplus (\mathbf{r}_t^* \odot \mathbf{h}_{t-1}^*)] + \mathbf{b}_h^*), \quad (4)$$

$$\mathbf{h}_t^* = (\mathbf{1} - \mathbf{z}_t^*) \odot \mathbf{h}_{t-1}^* + \mathbf{z}_t^* \odot \tilde{\mathbf{h}}_t^*, \quad (5)$$

where $\mathbf{W}_z^*, \mathbf{W}_r^*, \mathbf{W}_h^* \in \mathbb{R}^{d_h \times (d+d_h)}$ and $\mathbf{b}_z^*, \mathbf{b}_r^*, \mathbf{b}_h^* \in \mathbb{R}^{d_h}$ are parameters. $\mathbf{z}_t^*, \mathbf{r}_t^* \in \mathbb{R}^{d_h}$ are the update gate and the reset gate respectively. T-GRU shares the similar calculation process of a standard GRU.

The GRU that considers the attribute information is called A-GRU. The intuition behind A-GRU is that, for a node, the attribute information lies in both its node attributes and the representations of its node neighbors. Thus, to generate the new proposal to update the state vector, A-GRU will consider the neighborhood information besides the attributes of the node itself and the previous state vector (Eq. (9)). How to extract the neighborhood information will be discussed in Section III-B.

A-GRU is described in Eqs. (6)-(10). Given the node attribute vectors at different time steps $\mathbf{x}_1$, ..., $\mathbf{x}_T \in \mathbb{R}^d$ and the neighborhood representation vectors $\mathbf{e}_1$, ..., $\mathbf{e}_T \in \mathbb{R}^{d_g}$, a state vector $\mathbf{h}_t^{'} \in \mathbb{R}^{d_h}$ is learned for each time step by applying the following equations iteratively.

$$\mathbf{z}_t^{'} = \sigma(\mathbf{W}_z^{'}[\mathbf{x}_t \oplus \mathbf{h}_{t-1}^{'} \oplus \mathbf{e}_t] + \mathbf{b}_z^{'}), \quad (6)$$

$$\mathbf{r}_t^{'} = \sigma(\mathbf{W}_r^{'}[\mathbf{x}_t \oplus \mathbf{h}_{t-1}^{'} \oplus \mathbf{e}_t] + \mathbf{b}_r^{'}), \quad (7)$$

$$\mathbf{s}_t^{'} = \sigma(\mathbf{W}_s^{'}[\mathbf{x}_t \oplus \mathbf{h}_{t-1}^{'} \oplus \mathbf{e}_t] + \mathbf{b}_s^{'}), \quad (8)$$

$$\tilde{\mathbf{h}}_t^{'} = \tanh(\mathbf{W}_h^{'}[\mathbf{x}_t \oplus (\mathbf{r}_t^{'} \odot \mathbf{h}_{t-1}^{'}) \oplus (\mathbf{s}_t^{'} \odot \mathbf{e}_t)] + \mathbf{b}_h^{'}), \quad (9)$$

$$\mathbf{h}_t^{'} = (\mathbf{1} - \mathbf{z}_t^{'}) \odot \mathbf{h}_{t-1}^{'} + \mathbf{z}_t^{'} \odot \tilde{\mathbf{h}}_t^{'}, \quad (10)$$

where $\mathbf{b}_z^{'}$, $\mathbf{b}_r^{'}$, $\mathbf{b}_s^{'}$, $\mathbf{b}_h^{'} \in \mathbb{R}^{d_h}$, $\mathbf{W}_s^{'} \in \mathbb{R}^{d_g \times (d+d_h+d_g)}$, $\mathbf{W}_z^{'}$, $\mathbf{W}_r^{'}$, $\mathbf{W}_s^{'} \in \mathbb{R}^{d_h \times (d+d_h+d_g)}$ are the parameters. $\mathbf{z}_t^{'}$, $\mathbf{r}_t^{'} \in \mathbb{R}^{d_h}$, $\mathbf{s}_t^{'} \in \mathbb{R}^{d_g}$ are the update gate, the reset gate and the neighborhood gate respectively.

Eqs. (6)-(8) describe how to calculate the three gates. Eq. (9) describes how to calculate a new proposal. The values in all gates are in the range of [0,1] and they control the information when generating the state vector. $\mathbf{r}_t^{'} \odot \mathbf{h}_{t-1}^{'}$ indicates how much information to keep from the previous state vector. $\mathbf{s}_t^{'} \odot \mathbf{e}_t$ indicates how much information to keep from the neighborhood representation vector. Eq. (10) describes how to calculate a new state vector.

B. Vector Representation of the Neighborhood

A neighborhood vector is extracted for each node at each time step to represent its neighborhood information. The key idea is to aggregate the neighbors' representations. We consider K-hop neighbors.

First, we prepare the K-hop neighbors Given a set of nodes $\mathcal{B}$ to be classified, we sample the immediate neighbors of the nodes in $\mathcal{B}$. These sampled neighbors and the nodes in $\mathcal{B}$ together form a new set $\mathcal{B}^1$. We do the same for $\mathcal{B}^1$ and get another new set $\mathcal{B}^2$. As a result, we get a sequence of sets denoted by $\mathcal{B}^0$ ... $\mathcal{B}^K$, where $\mathcal{B}^0$ = $\mathcal{B}$. The neighbors are sampled by sampling function $\mathcal{N}(\cdot)$. Then, based on these node sets, we generate the neighborhood vectors for all the nodes in $\mathcal{B}$ (Algorithm 1). $\mathbf{g}_{t(v)}^k$ is the representation of node v at time step t after aggregating its k-th hop neighbors. Line 5 describes the operation of aggregating neighbors' representations. Line 6 describes how to generate a new representation. $\mathbf{W}_{trans}^k \in \mathbb{R}^{d_g \times 2d_g}$ is the transformation matrix that we need to learn. We will introduce aggregator $AGG(\cdot)$ in Section III-C1.

Algorithm 1: Generating Neighborhood Vector

Input: Temporal attributed graph $\mathbb{G} = (\mathcal{G}^1, \mathcal{G}^2, ..., \mathcal{G}^T)$, K-hop neighbors $\mathcal{B}_t^0$ ... $\mathcal{B}_t^K$, where $\mathcal{B}_t^0$ = $\mathcal{B}$.
Output: Neighborhood vector $\mathbf{e}_{t(v)}$ for all $v \in \mathcal{B}_t^0$.

1 **for** $t = 1, ..., T$ **do**
2 $\mathbf{g}_{t(v)}^K \leftarrow \mathbf{x}_{t(v)}, \forall v \in \mathcal{B}_t^K$
3 **for** $k = K$-1, ..., 1 **do**
4 **for** $v \in \mathcal{B}_t^k$ **do**
5 $\mathbf{g}_{\mathcal{N}(t(v))}^{k+1} \leftarrow AGG_{k+1}(\{\mathbf{g}_{t(u)}^{k+1}, \forall u \in \mathcal{N}(v)\})$
6 $\mathbf{g}_{t(v)}^k \leftarrow \sigma(\boldsymbol{W}_{trans}^{k+1}[\mathbf{g}_{t(v)}^{k+1} \oplus \mathbf{g}_{\mathcal{N}(t(v))}^{k+1}])$
7 **for** $v \in \mathcal{B}_t^0$ **do**
8 $\mathbf{e}_{t(v)} \leftarrow AGG_1(\{\mathbf{g}_{t(u)}^1, \forall u \in \mathcal{N}(v)\})$

C. Triple Attention Module

A triple attention module is developed to model three types of factors that influence the node representations in dynamic networks.

1) Spatial Attention: Different neighbors influence node presentations diversely. Attention technique can adaptively capture the pertinent information [14]. We design a spatial attention module to detect the important neighbors of a node.

The spatial attention is applied to the aggregator during the aggregation process (Line 5 in Algorithm 1). Based on the attention values, the aggregator sums up the neighbors' representations as follows.

$$AGG_k(\{\mathbf{g}_{t(u)}^k, \forall u \in \mathcal{N}(v)\}) = \sum_{u \in \mathcal{N}(v)} \beta_u^k \mathbf{V}_k \mathbf{g}_{t(u)}^k, \quad (11)$$

where $\sum \beta_u^k$ = 1 and $\mathbf{V}_k \in \mathbb{R}^{d_g \times d_g}$ are parameters. β_u^k is the attention value of neighbor u located at the k-th hop. It indicates the importance of u to node v compared to other neighbors located at the k-th hop. β_u^k is produced by our spatial attention module that takes the representations of the node and its neighbors as inputs, which is described as follows.

$$\beta_u^k = \frac{\exp\{F(\mathbf{w}_k^\top [\mathbf{V}_k \mathbf{g}_{t(u)}^k \oplus \mathbf{V}_k \mathbf{g}_{t(v)}^k])\}}{\sum_{v' \in \mathcal{N}(v)} \exp\{F(\mathbf{w}_k^\top [\mathbf{V}_k \mathbf{g}_{t(v')}^k \oplus \mathbf{V}_k \mathbf{g}_{t(v)}^k])\}}, \quad (12)$$

where $F(\cdot)$ is an activation function. $\mathbf{w}_k \in \mathbb{R}^{d_g}$ and $\mathbf{V}_k \in \mathbb{R}^{d_g \times d_g}$ are parameters.

2) Attribute-Topology Attention: Node representations can be dominated by the network topology or node attributes. The relative importance of topology and attributes can be different

at different time steps. Based on this insight, a attribute-topology attention module is designed to pay different levels of attention to the topology and attributes.

The attribute-topology attention takes the state vectors $\mathbf{h}_t^*$ and $\mathbf{h}_t^{'}$ as inputs and outputs the attention values γ_t^* and $\gamma_t^{'}$:

$$\gamma_t^* = \frac{\exp\{\dot{\mathbf{w}}^\top \tanh(\dot{\mathbf{V}}\mathbf{h}_t^*)\}}{\exp\{\dot{\mathbf{w}}^\top \tanh(\dot{\mathbf{V}}\mathbf{h}_t^*)\} + \exp\{\dot{\mathbf{w}}^\top \tanh(\ddot{\mathbf{V}}\mathbf{h}_t^*)\}}, \tag{13}$$

$$\gamma_t^{'} = \frac{\exp\{\dot{\mathbf{w}}^\top \tanh(\dot{\mathbf{V}}\mathbf{h}_t^{'})\}}{\exp\{\dot{\mathbf{w}}^\top \tanh(\dot{\mathbf{V}}\mathbf{h}_t^{'})\} + \exp\{\dot{\mathbf{w}}^\top \tanh(\dot{\mathbf{V}}\mathbf{h}_t^{'})\}}, \tag{14}$$

where $\dot{\mathbf{w}}^\top \in \mathbb{R}^{d_\gamma}$ and $\dot{\mathbf{V}} \in \mathbb{R}^{d_\gamma \times d_h}$ are parameters. γ_t^* and $\gamma_t^{'}$ represents the relative importance of topology and attributes respectively at time step t for influencing the target node's representation. Thus, the final state vector at time step t is

$$\mathbf{h}_t = [(\gamma_t^* \times \mathbf{h}_t^*)^\top \oplus (\gamma_t^{'} \times \mathbf{h}_t^{'})^\top]^\top \in \mathbb{R}^{2d_h}. \tag{15}$$

To study the overall relative importance of topology and attributes for each node another attribute-topology attention module is designed:

$$\gamma^* = \frac{\exp\{\ddot{\mathbf{w}}^\top \tanh(\ddot{\mathbf{V}}\mathbf{h}^*)\}}{\exp\{\ddot{\mathbf{w}}^\top \tanh(\ddot{\mathbf{V}}\mathbf{h}^*)\} + \exp\{\ddot{\mathbf{w}}^\top \tanh(\ddot{\mathbf{V}}\mathbf{h}^*)\}}, \tag{16}$$

$$\gamma^{'} = \frac{\exp\{\ddot{\mathbf{w}}^\top \tanh(\ddot{\mathbf{V}}\mathbf{h}^{'})\}}{\exp\{\ddot{\mathbf{w}}^\top \tanh(\ddot{\mathbf{V}}\mathbf{h}^{'})\} + \exp\{\ddot{\mathbf{w}}^\top \tanh(\ddot{\mathbf{V}}\mathbf{h}^{'})\}}, \tag{17}$$

where $\mathbf{h}^* = [\mathbf{h}_1^{*\top} \oplus \cdots \oplus \mathbf{h}_T^{*\top}]^\top \in \mathbb{R}^{Td_h}$ and $\mathbf{h}^{'} = [\mathbf{h}_1^{'\top} \oplus \cdots \oplus \mathbf{h}_T^{'\top}]^\top \in \mathbb{R}^{Td_h}$. $\ddot{\mathbf{w}}^\top \in \mathbb{R}^{d_\gamma}$ and $\ddot{\mathbf{V}} \in \mathbb{R}^{d_\gamma \times Td_h}$ are parameters. γ^* and $\gamma^{'}$ represent the overall relative importance of topology and attributes respectively. Thus, the final state vector based on this attention module is

$$\mathbf{h}_t = [(\gamma^* \times \mathbf{h}_t^*)^\top \oplus (\gamma^{'} \times \mathbf{h}_t^{'})^\top]^\top \in \mathbb{R}^{2d_h}. \tag{18}$$

3) Temporal Attention: For a dynamic network, the amount of valuable information provided by different time steps is different. Only some contain the most discriminative information for determining node representations. Thus, we design a temporal attention module to pay different levels of attention to different time steps.

The temporal attention module takes the state vector $\mathbf{h}_t$ as input and outputs an attention value:

$$\alpha_t = \frac{\exp\{\tilde{\mathbf{w}}^\top \tanh(\tilde{\mathbf{V}}\mathbf{h}_t)\}}{\sum_{i=1}^{T} \exp\{\tilde{\mathbf{w}}^\top \tanh(\tilde{\mathbf{V}}\mathbf{h}_i)\}}, \tag{19}$$

where $\tilde{\mathbf{w}} \in \mathbb{R}^{d_\alpha}$ and $\tilde{\mathbf{V}} \in \mathbb{R}^{d_\alpha \times 2d_h}$ are parameters. α_t indicates the importance of time step t for influencing the target node's representation compared to others. We concatenate all $\mathbf{h}_t$ as $\mathbf{H} = [\mathbf{h}_1 \oplus \cdots \oplus \mathbf{h}_T] \in \mathbb{R}^{T \times 2d_h}$. Therefore the attention values of different time steps are $\boldsymbol{\alpha} = \text{softmax}(\tilde{\mathbf{w}}^\top \tanh(\tilde{\mathbf{V}}\mathbf{H}^\top)) \in \mathbb{R}^T$. Then we sum up all the state vectors scaled by $\boldsymbol{\alpha}$ to generate the final vector representation for the node as $\mathbf{q} = \boldsymbol{\alpha}^\top \mathbf{H} \in \mathbb{R}^{2d_h}$.

D. Objective Function

Given the node representations denoted by $\mathbf{q}_1, \cdots, \mathbf{q}_N$ and the node labels $\mathbf{y}_1, \cdots, \mathbf{y}_N$, where N is the number of nodes, the objective function of AdaNN is

$$J = L_{ce} + \lambda P_{nn}. \tag{20}$$

TABLE I
DESCRIPTION OF THE DATASETS

Dataset	# Nodes	# Edges	# Attributes	# Time Steps	# Categories
Brain	5000	1955488	20	12	10
DBLP-5	6606	42815	100	10	5
Epinions	16025	1144258	20	11	10
Reddit	8291	264050	20	10	4

$L_{ce} = -\frac{1}{N} \sum_{i=1}^{N} \mathbf{y}_i \log(\tilde{\mathbf{y}}_i)$ is the cross-entropy loss. $\tilde{\mathbf{y}}_i$ is the estimate, which is produced by applying softmax$(\cdot)$ to the output of a fully connected layer that takes the node representation as input, i.e., $\tilde{\mathbf{y}}_i = \text{softmax}(\mathbf{W}_o\mathbf{q}_i + \mathbf{b}_o)$. $\mathbf{W}_o \in \mathbb{R}^{c \times 2d_h}$ and $\mathbf{b}_o \in \mathbb{R}^c$ are the parameters. c is the number of categories. P_{nn} is the penalization term for the parameters to prevent AdaNN from over-fitting. λ is a hyper-parameter.

IV. Experiments

A. Datasets

We use four real-world datasets. The description of datasets is shown in Table I. The datasets and the code of this paper are publicly available[1]. In the Brain dataset, nodes represent tidy cubes of brain tissue and edges indicate the connectivity. Brain is generated from the real-world task based functional magnetic resonance imaging (fMRI) data[2]. This fMRI data is collected when the subject conducts different tasks successively. We apply PCA to the fMRI of a time period to generate the node attributes for a network snapshot. Two nodes are connected if they show similar degree of activation during the time period. DBLP-5[3] is a co-author network dataset where the nodes represent the authors. The node attributes in a network snapshot are extracted from the titles and abstracts of the corresponding author's publications during a time period by word2vec. The authors in DBLP-5 are from five research areas. In Reddit[4], the nodes represent posts. We apply word2vec to the comments of a post to generate its node attributes [7]. Two nodes are connected in a network snapshot of if their corresponding posts contain similar keywords during a time period. Epinions is extracted from a product review site[5], where users construct trust graphs to seek advice from others. Nodes represent users and two nodes are connected in a network snapshot if one of them seeks advice from the other. Node attributes are generated from the reviews by word2vec.

B. Baseline Methods

We compare AdaNN with some baseline methods. DeepWalk [4], GAT [8], GCN [6], GraphSAGE [7] and node2vec [5] are originally designed for static networks. We apply them to each network snapshot to generate the node representation at each time step first. Then we concatenate these representations into a vector. A logistic regression is applied to predict the node label. GRU [12], DynGEM [10], DynAERNN [15],

[1]https://tinyurl.com/y6d74mmv
[2]https://tinyurl.com/y4hhw8ro
[3]https://dblp.uni-trier.de/
[4]https://www.reddit.com/
[5]http://www.epinions.com

TABLE II
NODE CLASSIFICATION COMPARISON (%).

Method	Brain			DBLP-5			Epinions			Reddit		
	ACC	AUC	F1	ACC	AUC	F1	ACC	AUC	F1	ACC	AUC	F1
DeepWalk	71.4	97.2	70.2	35.4	61.0	26.9	30.1	68.4	23.0	47.5	71.9	46.8
GAT	43.8	86.2	49.2	32.5	48.6	26.1	22.5	63.1	18.3	29.6	52.4	21.8
GCN	65.0	86.7	60.1	33.7	50.0	28.9	20.9	62.4	17.8	27.7	54.0	21.3
GraphSAGE	69.4	96.7	74.1	71.0	90.7	69.7	24.5	63.9	18.7	42.5	66.8	42.5
node2vec	71.0	96.8	70.6	36.9	64.2	27.2	32.8	70.2	26.0	48.0	72.2	47.9
GRU	80.4	98.2	80.2	75.6	91.5	75.2	17.3	61.7	17.2	42.1	67.2	41.9
DynGEM	49.6	90.6	50.0	41.0	66.0	22.3	32.7	66.8	24.1	27.1	51.4	17.3
DynAERNN	46.6	89.0	47.0	36.8	55.9	16.0	32.9	68.3	23.8	28.9	53.6	18.6
DANE	85.2	94.8	85.9	82.5	92.3	81.6	31.8	67.1	23.5	45.7	70.0	45.1
STAR	89.2	99.2	90.0	80.3	95.5	80.7	32.6	67.4	24.0	50.8	75.0	51.1
AdaNN-S	87.8	95.8	88.4	83.9	94.8	82.4	30.9	68.1	25.5	42.3	67.1	42.1
AdaNN-P	85.1	91.5	84.7	81.5	93.7	81.7	27.5	62.8	22.4	46.1	71.3	46.2
AdaNN-T	79.5	90.2	79.9	78.1	92.5	77.6	29.2	65.3	25.0	44.6	68.0	44.4
AdaNN	**91.0**	**99.5**	**92.3**	**88.5**	**97.8**	**88.4**	**34.5**	**72.7**	**28.1**	**52.1**	**79.6**	**53.4**

DANE [9] and STAR [16] are used to model the evolution of dynamic networks. GRU only consider the evolving patterns of node attributes. DynGEM and DynAERNN can model the evolving patterns of network topology. DANE learns offline node representations in terms of both network topology and node attributes. STAR employs a spatio-temporal GRU to jointly model temporal evolution of attributes and topology. To gain insights about AdaNN, we study its variants. AdaNN-S is the variant without applying the spatial attention. It generates neighborhood representations by the mean pooling. AdaNN-P is the variant without applying the attribute-topology attention. AdaNN-T does not apply the temporal attention.

In our experiments, λ is set to 10^{-3}. d_h, d_g, d_γ, d_α are set to 10. K is set to 2. The number of sampled neighbors n_s is set to 4. They are determined by grid-search from $\{0, 2\times10^{-4}, 5\times10^{-4}, 1\times10^{-3}, 2\times10^{-3}\}$, $\{2,5,10,15\}$, $\{1,2,3,4\}$ and $\{2,4,8,16\}$ respectively. $\mathcal{N}(\cdot)$ is the uniform sampling function. LeakyReLU is used as $F(\cdot)$. 10-fold cross-validation is applied. AdaNN is optimized by Adam.

C. Node Classification

Table II shows the results. It is observed that AdaNN achieves the best performance in general. DeepWalk and node2vec show good performance on the Epinions and Reddit datasets. This is because the representations of most nodes in the two datasets are dominated by network topology information, and DeepWalk and node2vec are good at extracting topology information. GRU shows good performance on Brain and DBLP-5. This is because a large part of node representations in the two datasets are dominated by the node attribute information and the temporal change. GRU has the advantage of modeling the evolving patterns of node attributes. DANE outperforms DynGEM and DynAERNN on Brain and DBLP-5 because it considers the evolving patterns of both attributes and topology. AdaNN outperforms STAR because it utilizes network topology better. AdaNN outperforms AdaNN-S, AdaNN-P and AdaNN-T, indicating the importance of the triple attention on learning better node representations for classification. AdaNN achieves good performance in general because it is adaptive to the dynamic characteristics of different networks and can model different factors for node representations.

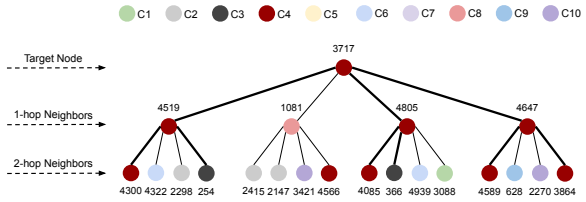


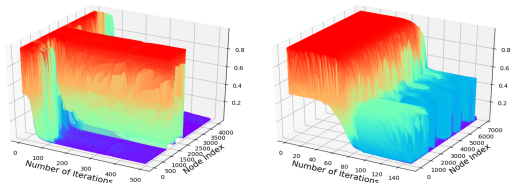
Fig. 3. Visualization of spatial attention values. Edge thickness indicates attention value between nodes.

D. Insights of Effectiveness

1) Visualization of Spatial Attention Values: To evaluate the spatial attention module, we visualize the attention values of different node neighbors. We take node 3717 from Brain at time step 10 as an example. The visualization is shown in Fig. 3. Different colors represent different categories. Edge thickness indicates attention value between nodes. Thicker edge represents larger value, which indicates that the neighbor is more important to the node. It is observed that for node 3717, the learned attention values of nodes 4519, 4805 and 4647 are larger. This is because these nodes are from the same category and show more influence on each other. For nodes 4519, 4805 and 4647, the neighbors from their category also have larger attention values. In addition, we observe the attention values of nodes 254 and 366 are also larger. This is because the two nodes are from the C3 category (Cognitive Control) that is functionally related to the C4 category (Body Movement) at time step 10. These observations verify the effectiveness of the spatial attention module.

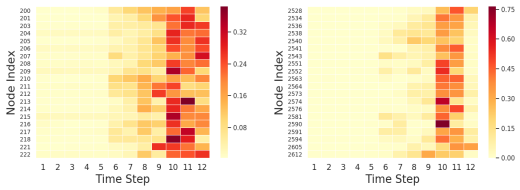
2) Visualization of Attribute-Topology Attention Values: To verify the effectiveness of the attribute-topology attention module, we visualize the attention paid to the node attribute information (the attention paid to the network topology information are the opposite). Fig. 4 shows the attention values for attribute information of each node with respect to different number of iterations on Brain and Reddit respectively. The red color represents higher attention value. It is observed that along the iterations, the attention values for attribute information evolve differently for the Brain and Reddit nodes. This is because the network properties of different networks are different. This is because the network properties of different node categories in the Brain network are different, but are very similar in the Reddit network. These observations verify the effectiveness and the necessity of the attribute-topology attention module that focuses on the overall network property.

3) Visualization of Temporal Attention Values: To evaluate the temporal attention module, we visualize the attention values of different time steps. Fig. 5 shows the visualization results of two categories of nodes from Brain. The red color represents higher attention values. It is observed that different categories of brain nodes have different attention value distribution over time steps and the nodes from the same category share the similar attention distribution. This is



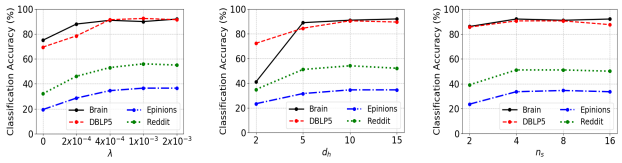
(a) All nodes of Brain (b) All nodes of Reddit

Fig. 4. Visualization of attention paid to attribute information.



(a) Body Movement (b) Cognitive Control

Fig. 5. Visualization of temporal attention at different time steps of different categories of brain nodes.



(a) Varying λ (b) Varying d_h (c) Varying n_s

Fig. 6. Performance evaluation of AdaNN.

because the subject conducts different tasks along the time, and different categories of brain nodes show different degrees of activation under different tasks. These observations verify the effectiveness of the temporal attention module.

4) Parameter Sensitivity Study: We study the sensitivity of AdaNN with respect to three parameters, the hyper-parameter λ, the state vector size d_h, and the number of sampled neighbors n_s. Figs. 6(a)-6(c) show the classification accuracy on four datasets by changing one parameter while fixing others as the same as Section IV-B. It is observed that as these parameters increase, the accuracy increases in general until an optimal value. $\lambda \in [4\times10^{-4}, 1\times10^{-3}]$, $d_h \in$ [10,15], and $n_s \in$ [4,8] give the optimal results. Thus it is reasonable to set them as 10^{-3}, 10 and 4 respectively. Moreover, the non-zero choices of λ verify the importance of the penalization term.

V. Related Work

Node classification has drawn extensive research attention in recent years [4], [6]–[8]. Perozzi et al. in [4] apply the random walks to extract the local information of a node in networks. Nodes are then classified by the logistic regression model. Veličković et al. in [8] apply attention mechanism to perform node classification in networks. These methods are for the static networks. However, many real-world networks are dynamic. Both network topology and node attributes evolve over time. Some progresses have been made on the node classification in the dynamic networks recently. Zhou et al. in [11] use the triadic closure process to model the evolution patterns of dynamic networks and a logistic regression model is applied to classify nodes. Goyal et al. in [10] propose a node embedding algorithm based on autoencoder, which builds the embedding of snapshot at time t from the embedding at time t-1. However, none of these approaches can be adaptive to the dynamic characteristics of different networks and model the different factors that influence node representations.

VI. Conclusion

Node classification targets to classify nodes in networks where labels are known for only a subset of nodes. Many real-world networks are dynamic. In this paper, we propose AdaNN for node classification in dynamic networks. AdaNN consists of two connected GRUs and a triple attention module. The connected GRUs take node attribute information and network topology information as inputs to generate the spatio-temporal contextual information. The triple attention module is designed to automatically model different factors that influence node representations. Extensive experimental results demonstrate the effectiveness of AdaNN.

Acknowledgements

This work was partially supported by the National Science Foundation grant IIS-1707548.

References

[1] S. Wang, L. He, B. Cao, C.-T. Lu, P. S. Yu, and A. B. Ragin, "Structural deep brain network mining," in *SIGKDD*. ACM, 2017, pp. 475–484.
[2] J. Ni, S. Chang, X. Liu, W. Cheng, H. Chen, D. Xu, and X. Zhang, "Co-regularized deep multi-network embedding," in *WWW*. International World Wide Web Conferences Steering Committee, 2018, pp. 469–478.
[3] Z. Bai, P. Walker, and I. Davidson, "Mixtures of block models for brain networks," in *SDM*. SIAM, 2018, pp. 46–54.
[4] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *SIGKDD*. ACM, 2014, pp. 701–710.
[5] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *SIGKDD*. ACM, 2016, pp. 855–864.
[6] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.
[7] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NeurIPS*, 2017, pp. 1024–1034.
[8] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," 2018.
[9] J. Li, H. Dani, X. Hu, J. Tang, Y. Chang, and H. Liu, "Attributed network embedding for learning in a dynamic environment," in *CIKM*. ACM, 2017, pp. 387–396.
[10] P. Goyal, N. Kamra, X. He, and Y. Liu, "Dyngem: Deep embedding method for dynamic graphs," *arXiv preprint arXiv:1805.11273*, 2018.
[11] L. Zhou, Y. Yang, X. Ren, F. Wu, and Y. Zhuang, "Dynamic network embedding by modeling triadic closure process," in *AAAI*, 2018.
[12] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," in *EMNLP*, 2014.
[13] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *AAAI*, 2016.
[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017, pp. 5998–6008.
[15] P. Goyal, S. R. Chhetri, and A. Canedo, "dyngraph2vec: Capturing network dynamics using dynamic graph representation learning," *arXiv preprint arXiv:1809.02657*, 2018.
[16] D. Xu, W. Cheng, D. Luo, X. Liu, and X. Zhang, "Spatio-temporal attentive rnn for node classification in temporal attributed graphs," in *IJCAI*, 2019, pp. 3947–3953.