A Strong Node Classification Baseline for Temporal Graphs

Farimah Poursafaei *‡ farimah.poursafaei@mila.quebec
Zeljko Zilic * zeljko.zilic@mcgill.ca
Reihaneh Rabbany †‡ rrabba@cs.mcgill.ca

Abstract

Many real-world complex systems can be modelled by temporal networks. Representation learning on these networks often captures their dynamic evolution and is a first step for performing further analysis, e.g. node classification. Node classification is a fundamental task for graph analysis in general and in the context of temporal graph, is often employed to categories nodes based on their activity patterns. Analysis of existing real world networks from different high-stake domains reveals that the rate of the malicious activities is on uptick, resulting in catastrophic social or economic consequences. This strongly motivates designing accurate node classification methods for temporal graphs.

In this paper, we propose TGBASE, for node classification on weighted temporal networks. TGBASE *efficiently* extracts key features to consider the structural characteristics of each node and its neighborhood as well as the intensity and timestamp of the interactions among node pairs. These features *accurately* differentiate different classes of nodes, as shown on eight real-world benchmark datasets, outperforming multiple state-of-the-art (SOTA) deep/complex models. Our strong yet simple model is also *generic*, whereas the SOTA contenders are designed often for their specific (class of) datasets.

1 Introduction

Various real-world complex systems can be abstracted by temporal networks that describe relations or interactions. Temporal networks have ubiquitous applicability in wide range of domains such as cryptocurrency transactions networks [22], social networks [23], and recommendation systems [31, 12]. Recently, extensive research has been conducted over graph structured data to learn vector representations of network elements such as nodes or interactions [23, 31, 25, 27, 26, 13]. Although temporal evolution is a principal property of many applications, research has mainly focused on static graphs where network elements or their attributes are fixed over time [7]. In representation learning on temporal graphs, it is important to note that topological structure as well as node and edge features are evolving over time [31]. Therefore, it is necessary to efficiently integrate temporal, structural, and semantic characteristics of the networks elements.

A fundamental task in network analysis is node classification where the objective is to categorize the node into one of the predefined classes [2, 7]. Node classification has many practical applications on real-world networks. In particular, it can be adopted to predict the authenticity of network users to prevent illicit activities. In fact, as a side effect of the relentless growth of various networks, increasing opportunities exist for the malicious parties to take advantage of manipulating the network data in different context: from cryptocurrency transaction networks (e.g. scammers), to social network (e.g. trolls), to e-commerce interaction networks (e.g. fraudsters) and many more [21]. For instance, in e-commerce networks where consumers make decisions based on the user-generated contents, such as ratings and reviews, there is a financial motivation to manipulate the network by fake ratings [10, 9]. Similarly, rating platforms also exist in the context of cryptocurrency transaction networks, where participants examine reviews and ratings of an online service or product as a measure of trust to decide about whom to trade cryptocurrency with. Thus, fraudulent users find significant incentives to give fake ratings [11].

Considering the huge economical and social impacts of conducting malicious activities on large networks, considerable attention has been given to protecting the networks [21]. In particular, a large body of data mining and machine learning techniques have been developed for examining the interactions of the users within a network and discovering potentially malicious activities. Among them, the most effective ones are the graph-based approaches such as [10, 9, 30, 28], which look at the activity patterns summarized as a network structure. Here, we follow this graph-based line of work and propose a strong baseline for node classification in temporal graph that outperforms more expensive state-of-the-art contenders (see Table 1).

In this paper, we model online interaction networks, including cryptocurrency transaction networks, rating networks, and social networks, as weighted temporal graphs. In these commonplace graphs, there is a time associated with each interaction between users, as well as an intensity (or weight) and direction. Our proposed approach, TGBASE, incorporates interaction-based, as well as structural and semantic attributes of the nodes to set out a carefully-designed set of features for each node with no learning needed or parameters to adjust. TGBASE provides simple yet effective node representations that can be employed for distinguishing different types of nodes. We compare TGBASE with many state-of-the-art models which learn representations from the data often using complex deep models. We show that our shallow model

*ECE, McGill University, Montreal, Canada
†CSC, McGill University, Montreal, Canada
‡Mila, Montreal, Canada

	NE*		GNN†			TGRL‡				Platform-dependent			
	node2vec [5]	RiWalk [15]	GCN [8]	GraphSAGE [6]	GAT [26]	JODIE [12]	DyRep [25]	TGAT [31]	TGN [23]	Weber et al. [28]	trans2vec [30]	REV2 [10]	**TGBASE**
Uses network structure	✔	✔	✔	✔	✔		✔	✔	✔		✔	✔	✔
Uses interaction information						✔	✔	✔	✔	✔	✔	✔	✔
Considers network dynamics						✔	✔	✔	✔		✔		✔
Parameter free										✔		✔	✔
Scalable				✔	✔	✔	✔	✔	✔				✔
Generic	✔	✔	✔	✔	✔	✔	✔	✔	✔				✔

Table 1: TGBASE meets all desirable properties. *: Network embedding methods. †: Graph neural networks methods. ‡: Temporal graph representation learning methods.

which incorporates these effective features outperforms these contenders while being simpler and more efficient for node classification task in *both* static and dynamic setting. The main contributions of our paper are three-fold:

- We propose an *efficient* method, TGBASE, for node classification in temporal interaction graphs.
- We conduct extensive experiments, on large scale graphs with millions of edges, to show TGBASE is *accurate*, *scalable* and *general*.
- We release TGBASE as a simple baseline for node classification in temporal graphs. The TGBASE-features are simple and easy to generate.

Reproducibility: Our code is open-sourced at `https://github.com/fpour/TGBase`. This includes the datasets and scripts for reproducing the reported results.

2 Related Work

We classify the existing works on learning representations for node classification into the following themes:

2.1 Network Embedding. These methods aim at developing a mapping function from a discrete graph to a continuous domain [7]. The goal is to learn a vector representation for each node such that important global or local properties are preserved. As pioneering methods, *deepwalk* [20] and *node2vec* [5] employ random walks to capture the structure of the given graph which are fed into skip-gram architecture [17] to generate embeddings that places similar nodes close together. With the biased random walks, node2vec embeddings can interpolate between embeddings based on community structure and structural roles. More recently, *RiWalk* [15] proposes to relabel nodes based on their structural roles before generating the random walks to further improve the embeddings. These shallow unsupervised embeddings models are outperformed by more powerful graph neural networks specifically for node classification where it might be better to learn the task end-to-end, e.g. generate embeddings specifically suited for the node classification.

2.2 Graph Neural Network (GNN). These methods mainly employ message passing procedures over the graph to generate node representations [13] by feature smoothing over the local neighbourhoods, where the representations are initialized with the explicit node attributes/features when available [13, 8, 6, 26]. *GCN* [8] is a pioneering model that learns node representations for the semi-supervised node classification by deriving the importance of each neighbour in the aggregate function directing from a normalized adjacency matrix. *GAT* [26] is an attention-based variation of GCN where at every layer, learns the importance of neighbor nodes instead of deriving it. *GraphSAGE* [6] reduces the computational complexity of GCN by sampling the neighbourhoods. Although the literature on GNNs is vast, there are relatively few works that explore extensions to dynamic graphs.

2.3 Temporal Graph Representation Learning. For modeling dynamic graphs, several methods employ RNNs as a common choice for extending the sequence models to temporal graphs [7]. *JODIE* [12] focuses on the temporal interactions of the users and items in social networks and learns the node embeddings through coupled RNNs. *DyRep* [25] is a model that update the node representations upon observing a new event in the network via its custom RNN. *TGAT* [31] propagates messages from sampled neighbors of a specific node similar to GNN methods. The sampling strategy of TGAT requires saving historical neighbors, which may hinder the method from being adopted in online learning. *TGN* [23] combines graph-based and memory operators for presenting an efficient and generic framework for learning on temporal graphs. Rossi et al. [23] demonstrate that TGN is a generalization of several models for learning on temporal graphs which are presented as a sequence of events.

2.4 Platform-dependent Methods. Many algorithms are introduced for incorporating network structure into node classification in a specific domain [10, 29, 30, 14, 32, 22]. Notably for rating networks, *REV2* [10] models the network as a signed network and learns node representations to detect fraudsters by incorporating specific behavioral attributes. For blockchain-based cryptocurrency networks, recent works in [14, 29, 32, 30] propose specific modes to detect different types of malicious activities on the Ethereum network. In particular, *trans2vec* [30] inspects Ethereum transactions by modeling the network as a multidirectional graph, where edge weights are based on a combination of amount and time interval of the transactions, and node representations are learned through a random walk-based method. [4, 3] propose features for the Ethereum accounts addresses based on their transaction histories to indirectly capture the network characteristics. Similarly for Bitcoin network, many recent works [28, 1, 19, 18, 24] extract explicit features to detect various anomalous activities such as phishing, money laundering, and ransomware attacks. In particular, Weber et al. [28] focus on the Bitcoin transaction network and extract a set of features which consists of *local* features (such as average Bitcoin sent/received or transaction fees) and *aggregated* transaction features considering first-order neighbors of the target nodes. In SIGTRAN [22], we propose a generic sets of features which is combined with node embeddings to detect illicit nodes on instances of the Bitcoin as well as Ethereum network. The work in this paper is built upon this. Here, we show an efficient set of features generalizes across different domains, and is able to perform competitively in node classification of graphs from different domains. Our TGBASE provides an efficient and strong baseline for node classification in weighted temporal networks. Table 1 provides an overview comparison of TGBASE with important related works.

3 Proposed Method

We devise comprehensive node encodings to represent semantic, structural, and temporal attributes of each node in a temporal network useful to distinguish different types of nodes. The network is modeled as a weighted temporal graph $G = (V, E)$, where V and E denote the sets of nodes and edges, respectively. For each edge $e_{uv} \in E$ from node $u \in V$ to node $v \in V$, we denote its intensity by $w(e_{uv})$, and its timestamp by $t(e_{uv})$.

3.1 TGBASE. For each node u in a given weighted temporal directed graph, we extract a set of features. The TGBASE features include two main types (see Fig. 1): i), the ***structural*** attributes of node u (*self*) and its neighboring nodes (*neighborhood*); ii) the ***interaction***-*based* features that specify the *intensity* and *timestamp* of interactions between u and its neighbors. More specifically we have:

• *Structure-**Self*** features, which consist of in-degree, out-degree, and total degree of u. More formally defined as: $D_{in}(u) = \sum_{v \in V} |e_{vu}|$, $D_{out}(u) = \sum_{v \in V} |e_{uv}|$, and $D_{tot}(u) = D_{in}(u) + D_{out}(u)$, respectively.

• *Structure-**Neighborhood*** features, which aggregate the structural attributes of the neighbors of node u. We consider a set of six aggregation functions, denoted by A, consisting of *average*, *maximum*, *minimum*, *summation*, *standard deviation*, and *entropy* over an arbitrary distribution, denoted by x. More specifically:

$$A = \{\overline{x}, \max(x), \min(x), \sum(x), \sigma(x), H(x)\} \tag{3.1}$$

Given the first-order neighbors of node u, $N_u = \{v \,|\, e_{uv} \in E \vee e_{vu} \in E\}$, we first derive the three structural attributes of each neighbour v, i.e. $[D_{in}(v), D_{out}(v), D_{tot}(v)]$, and then apply the six aggregation functions in (3.1) over each of these three attributes to get our 18 aggregated neighborhood features for u. Table 2 provides the complete list of the resulted features.

• *Interaction-**Intensity*** features express the aggregated statistics of the intensity of interactions among node u and its neighbours. We consider three sets of edges, incoming, outgoing and total edges, and aggregate their intensity with the same aggregation functions in (3.1) to gain 18 edge-weight or intensity related features. More specifically: $W_{in}(u) = \{g(w_u^{in}) \mid g \in A, w_u^{in} = \{w(e_{vu}) \mid v \in N_u\}\}$, $W_{out}(u) = \{g(w_u^{out}) \mid g \in A, w_u^{out} = \{w(e_{uv}) \mid v \in N_u\}\}$, and $W_{tot}(u) = \{g(w_u^{tot}) \mid g \in A, w_u^{tot} = \{w(e_{vu}), w(e_{uv}) \mid v \in N_u\}\}$. This generalizes to multi-graphs, where there may exist several edges among each pair of nodes, since it is inherently summarizing the characteristics of multiple interactions.

• *Interaction-**Timestamp*** features express the aggregated statistics of interval of interactions among node pairs. Considering the interaction time interval of node u and its neighbor v together with aggregation functions in (3.1), the timestamp features of node u are acquired as follows: $T_{in}(u) = \{g(t_u^{in}) \mid g \in A, t_u^{in} = \{t(e_{vu}) \mid v \in N_u\}\}$, $T_{out}(u) = \{g(t_u^{out}) \mid g \in A, t_u^{out} = \{t(e_{uv}) \mid v \in N_u\}\}$, and $T_{tot}(u) = \{g(t_u^{tot}) \mid g \in A, t_u^{tot} = \{t(e_{vu}), t(e_{uv}) \mid v \in N_u\}\}$.

The aforementioned features are summarized in Table 2. In total, we defined only 57 local features per node. It is worth noting that for preserving semantic information of the nodes, we concatenate nodes initial attributes with the TGBASE features in case the graphs are attributed, i.e. explicit features are provided for nodes.

3.2 Static vs. Dynamic Node Classification. TGBASE can provide node representations for *static* as well as *dynamic* node classification. Traditionally, node classification on graphs have been considered in a static setting, meaning that the categories of the nodes are fixed, and based on the observed interactions of the nodes, we pre-

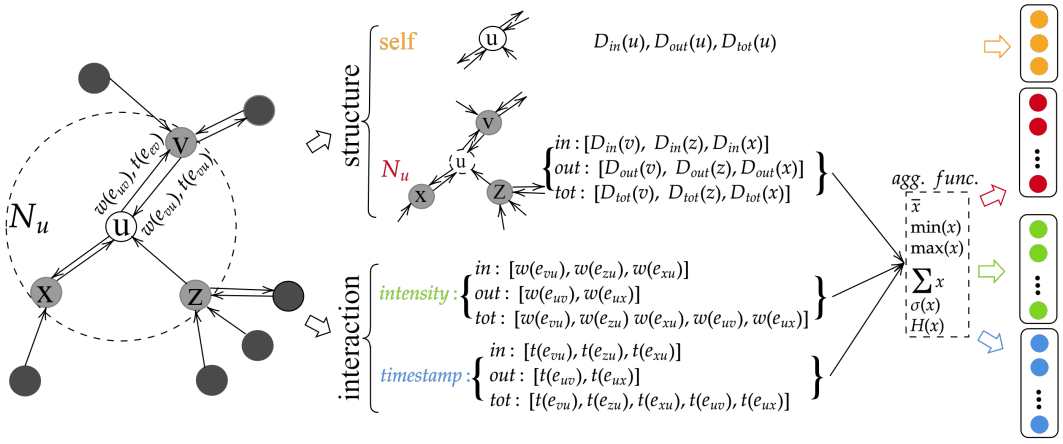

Figure 1: TGBASE features for a given node u incorporates two types of local features: *structure*-based features (*self* and *neighborhood*), and *interaction*-based features (*intensity* and *timestamp*).

Table 2: Full list of TGBASE features.

	interaction-based		structure-based	
	intensity (w)	time (t)	neighborhood (N_u)	self (u)
target	$min(w_u^{in})$ $max(w_u^{in})$ $\overline{w_u^{in}}$ $\sum w_u^{in}$ $\sigma(w_u^{in})$ $H(w_u^{in})$	$min(t_u^{in})$ $max(t_u^{in})$ $\overline{t_u^{in}}$ $\sum t_u^{in}$ $\sigma(t_u^{in})$ $H(t_u^{in})$	$min(D_{in}(N_u))$ $max(D_{in}(N_u))$ $\overline{D_{in}(N_u)}$ $\sum D_{in}(N_u)$ $\sigma(D_{in}(N_u))$ $H(D_{in}(N_u))$	$D_{in}(u)$
source	$min(w_u^{out})$ $max(w_u^{out})$ $\overline{w_u^{out}}$ $\sum w_u^{out}$ $\sigma(w_u^{out})$ $H(w_u^{out})$	$min(t_u^{out})$ $max(t_u^{out})$ $\overline{t_u^{out}}$ $\sum t_u^{out}$ $\sigma(t_u^{out})$ $H(t_u^{out})$	$min(D_{out}(N_u))$ $max(D_{out}(N_u))$ $\overline{D_{out}(N_u)}$ $\sum D_{out}(N_u)$ $\sigma(D_{out}(N_u))$ $H(D_{out}(N_u))$	$D_{out}(u)$
either	$min(w_u^{tot})$ $max(w_u^{tot})$ $\overline{w_u^{tot}}$ $\sum w_u^{tot}$ $\sigma(w_u^{tot})$ $H(w_u^{tot})$	$min(t_u^{tot})$ $max(t_u^{tot})$ $\overline{t_u^{tot}}$ $\sum t_u^{tot}$ $\sigma(t_u^{tot})$ $H(t_u^{tot})$	$min(D_{tot}(N_u))$ $max(D_{tot}(N_u))$ $\overline{D_{tot}(N_u)}$ $\sum D_{tot}(N_u)$ $\sigma(D_{tot}(N_u))$ $H(D_{tot}(N_u))$	$D_{tot}(u)$

dict the node classes. Therefore, it suffices to construct only one representation per node to be exploited by the downstream classification task. In this setting, we construct TGBASE representations only once based on the observed history of the evolution of the graph.

However, the evolution of the temporal graphs can be considered as a sequence of timestamped events where at each timestamp, a new decision regarding the classes of the nodes should be made. This dynamic node classification setting implies that the label of the nodes may change over time and the goal is to predict the correct label/category of the nodes at specific timestamps. It is important to note that in the dynamic node classification, we need to update node representations after observation of an event (such as evolving connectivity or features over time) to keep the most up to date representation. Moreover, in contrast to static node classification where nodes categories are predicted only once, in dynamic setting, classification can happen after each event to evaluate the impact of the network evolution on nodes' categories. In dynamic node classification, after observation of each event, TGBASE updates representation of the affected nodes to reflect the alteration. The update procedure for all the features is straightforward. For node classification, TGBASE always utilizes the most up to date representation of the nodes based on the observed history so far.

4 Experiments

This section presents the evaluation of TGBASE for static and dynamic node classification task. We investigated eight real-world datasets presented in Table 3 in the context of two tasks, static (Section 4.1) and dynamic (Section 4.3) node classification. The first six rows of Table 3 provide information of the datasets for static node classification where the node classes are fixed, and the last two rows present the datasets utilized for dynamic node classification where node states may change over time. We explain the two settings and their corresponding datasets and baselines in the following sections.

4.1 TGBASE for Static Node Classification.

Here, we evaluate TGBASE for static node classification in temporal weighted networks.

Datasets. We consider the following available benchmark datasets:

• *Bitcoin.* We adopted Bitcoin transactions dataset presented by Weber et al. [28] that consists of more than $200K$ transactions of which 2% are labeled as *illicit* (cor-

Table 3: Statistics of the different benchmark datasets used for performance evaluation.

Dataset	Nodes	Edges	Benign nodes	Illicit nodes	Avg. degree	Node States	Network Category
Bitcoin	203,769	234,355	42,019	4,545	1.3002	static	cryptocurrency
Ethereum	2,973,489	13,551,303	2,972,324	1,165	9.1148	static	cryptocurrency
alpha	3,783	14,124	138	102	7.4671	static	cryptocurrency rating
OTC	5,881	21,492	136	180	7.3090	static	cryptocurrency rating
Amazon	330,317	560,804	2,358	241	3.3956	static	general rating
Epinions	195,805	4,835,208	9,291	1,013	42.7914	static	general rating
Reddit	10,000	672,447	10,634	366	15.6986	dynamic	social network
Wikipedia	8,227	157,474	8,010	217	4.4327	dynamic	social network

responding to different categories of malicious activities including scams, ransomware, etc.), 21% are *licit* (corresponding to transactions of genuine categories including miners, exchanges, etc.), and there are no ground truth labels available for the rest of the transactions. Weber et al. [28] have also shared their defined set of features. The dataset is fully anonymized and the intensity and timestamp of the edges are not available. Therefore, we are only able to extract *structural* features (namely *self* and *neighborhood*). To preserve the semantic attributes of the nodes, we combined features provided by Weber et al. [28] with TGBASE features.

• *Ethereum.* Transaction records of Ethereum network are presented by Wu et al. [30]. This dataset includes the transaction histories of different Ethereum accounts where 1165 accounts were involved in phishing activities. No ground truth labels are available for other nodes which are therefore assumed to be genuine.

• *OTC* and *Alpha.* These datasets consist of the user-to-user trust network of Bitcoin clients using OTC and Alpha platform for trading cryptocurrencies [10]. In these platforms, users who are anonymously trading Bitcoin can rate other members of the network based on their trustworthiness. Thus, fraudulent users have high monetary incentive for giving fake ratings. The ground truth in these networks are set based on the platform's founder rating. The founder and all other users he/she has rated highly positive are considered as benign, and the users who have been negatively rated by the benign users are marked fraudulent.

• *Amazon.* This is a user-to-product rating network [10]. The helpfulness of a rating can be a good indicative of a fraudulent behavior, thus users with equal to or more than 50 votes, where the fraction of helpful-to-total votes is ≥ 0.75, are considered as genuine, while if the same fraction is ≤ 0.25 the user is considered as fraudulent.

• *Epinions.* This dataset consists of a user-to-post network in which rating values varies in $[1, 6]$. A user-to-user trust network is used for defining the ground truth, where a user is labeled as fraudulent if its total trust rate is ≤ -10, and genuine if its trust rate is $\geq +10$ [16, 10].

Baselines. To make a comprehensive performance evaluation, we considered following baselines:

• *Network Embedding Baselines: node2vec* [5] *and RiWalk* [15]. We set the parameters of the node2vec in line with [5]: context size $k = 10$, embedding size $d = 64$, walk length $l = 5$, and number of walk per node $r = 20$. In addition, for better exploitation of the structural equivalency of the nodes, we set $p = 0.25$ and $q = 4$. We also used *RiWalk-WL* which captures the fine-grained similarities by imitating the neighborhood aggregation of the Weisfeiler-Lehman graph kernels [15].

• *GNN Baselines:* We compared the performance of TGBASE against three GNN baselines which offer end-to-end node classification: *GCN* [8], *GraphSAGE* [6], and *GAT* [26]. The GNN baselines contain multiple layers where at each layer, the input is the node representations at that layer and the output is the transformed representations. At the final layer, the node representations are utilized for the classification task. We implemented all the GNN baselines in PyTorch Geometric with three layers and weighted loss function to considered the class imbalance of the datasets. We considered two different cases for initializing the node features for GNN baselines. In the first scenario, GNN models exploited one-hot encoding of the nodes as the initial features, which is a common practice. In the second scenario, TGBASE is exploited for generating the initial features (indicated as *"*TGBASE $\rightarrow$ *GNN"* in Table 4).

• *Platform-dependent Baselines.* We also compared the performance of TGBASE with state of the art (SOTA) baselines that are specifically designed for a particular type of network. These methods rely on particular characteristics of the network for which they are developed.

◦ *Cryptocurrency networks.* We considered the method proposed by Weber et al. [28] and trans2vec [30] as baselines for Bitcoin and Ethereum network, respectively. We set the parameters of the trans2vec similar to those of node2vec for a fair comparison. Please note that these methods cannot be applied to a different domain or even cryptocurrency platform. For example, trans2vec is not applicable to Bitcoin and is defined only for Ethereum.

Table 4: Comparing the performance of TGBASE in terms of AUC with SOTA baselines in *static* node classification task. The **first**, **second**, and **third** best performing method are colored correspondingly.

Method	Bitcoin	Ethereum	Alpha	OTC	Amazon	Epinions
Platform-dependent	**0.955** ±0.006	0.827 ±0.010	0.833 ±0.112	0.873 ±0.104	0.847 ±0.155	0.856 ±0.112
GCN	0.545 ±0.056	0.556 ±0.069	0.729 ±0.042	0.697 ±0.036	0.610 ±0.011	0.534 ±0.071
GraphSAGE	0.553 ±0.054	0.647 ±0.145	0.624 ±0.101	0.681 ±0.114	0.571±0.051	0.560 ±0.069
GAT	0.582 ±0.005	0.576 ±0.103	0.732 ±0.081	0.836 ±0.033	0.513 ±0.019	0.521 ±0.031
TGBASE → GCN	0.901 ±0.007	0.792 ±0.020	0.912 ±0.023	0.942 ±0.019	0.934 ±0.020	0.859±0.011
TGBASE → GraphSAGE	**0.970** ±0.003	**0.958** ±0.010	0.984 ±0.003	0.983 ±0.004	0.930±0.012	**0.950** ±0.008
TGBASE → GAT	0.928 ±0.010	0.709 ±0.041	0.926 ±0.024	0.948 ±0.012	0.926 ±0.015	0.555 ±0.156
node2vec	0.669 ±0.012	0.817 ±0.026	**0.990** ±0.003	**0.996** ±0.001	**1.000** ±0.000	*nc*
RiWalk + RF	0.636 ±0.050	0.805 ±0.022	**0.991** ±0.003	**0.995** ±0.002	**1.000** ±0.000	*nc*
TGBASE	**0.953** ±0.006	**0.906** ±0.010	**0.999** ±0.001	**1.000** ±0.000	**1.000** ±0.000	**0.999** ±0.000
node2vec	0.636 ±0.020	0.793 ±0.026	0.716 ±0.071	0.784 ±0.046	**0.999** ±0.001	*nc*
RiWalk + MLP	0.626 ±0.055	0.722 ±0.036	0.681 ±0.068	0.770 ±0.042	**0.999** ±0.001	*nc*
TGBASE	0.940 ±0.008	**0.892** ±0.014	0.764 ±0.102	0.892 ±0.0.052	**0.936** ±0.017	**0.932** ±0.010

◦ *Cryptocurrency rating and general rating networks.* We considered REV2 [10], which is the SOTA method for detecting fraudsters cryptocurrency rating benchmarks.

Performance Evaluation. In case of Bitcoin and Ethereum networks, we considered the malicious nodes and an equal number of genuine nodes as the set of anchor nodes and then, extracted the first-order neighbors of the anchor nodes and all the edges amongst them to extract subgraphs from the original networks. We repeated the random genuine anchor nodes selection procedure 10 times, and reported the average performance. For the rating networks, namely Alpha, OTC, Amazon, and Epinions we considered the classification task to be applied on all labeled nodes, while the available graphs are utilized in generating node representations. We reported the average performance over 10 different iterations of random train-test splits. The performance is measured using *area under the ROC curve (AUC)*.

After generating node representations with TGBASE, we need a classifier for predicting the category of the nodes. We considered two different classifiers. Namely, we implemented a three-layered *Multi-Layer Perceptron (MLP)* classifier with a ReLU activation function in PyTorch. We also exploited *Random Forest (RF)* due to its high performance [28, 10]. RF (with the following setup: number of estimator $= 50$, maximum number of features $= 10$, and maximum depth $= 5$) was implemented using Scikit-learn Python package. The evaluation results are illustrated in Table 4.

Results. We discuss the results of each dataset in the following.

• *Bitcoin.* The results in the second column of Table 4 show that both generic node embedding methods and GNNs (with one-hot encodings) achieve a relatively low performance on this benchmark. This is expected since these models do not deal with the intrinsic characteristics of financial networks, and mainly emphasize on the structural similarities to gain the node embeddings. However, TGBASE which is also a generic method performs competitively to the platform dependent contender on this dataset. This can be attributed to the fact that TGBASE takes into account important characteristics of the network mainly the interaction intensity and timestamp, which are overlooked by the baselines.

• *Ethereum.* The results in the third column of Table 4 suggest that TGBASE significantly outperforms the baselines in classification of the malicious nodes on the Ethereum network. Notably, TGBASE shows higher performance compared to trans2vec that is specifically designed for Ethereum network. The generic node embedding methods also give a relatively good performance on this dataset which suggests the structural information are important. The GNNs, when do not exploit TGBASE features, are however still underperforming on this dataset.

• *Alpha* and *OTC.* The results in the fourth and fifth column of Table 4 represent the higher performance of TGBASE compared to the baselines for OTC and Alpha datasets, respectively. Again, TGBASE achieves better performance compared to REV2, the domain specific SOTA, demonstrating that TGBASE's representations are more suitable for distinguishing malicious nodes. It is important to note that the size of the Alpha and OTC are much smaller than the size of the Bitcoin and Ethereum networks. This might explain why node2vec and RiWalk show better performacne on Alpha and OTC networks, compared to bigger networks. TGBASE can achieve good performance regardless of the network size.

• *Amazon.* The evaluation results are presented in the sixth column of Table 4. We observe a similar pattern

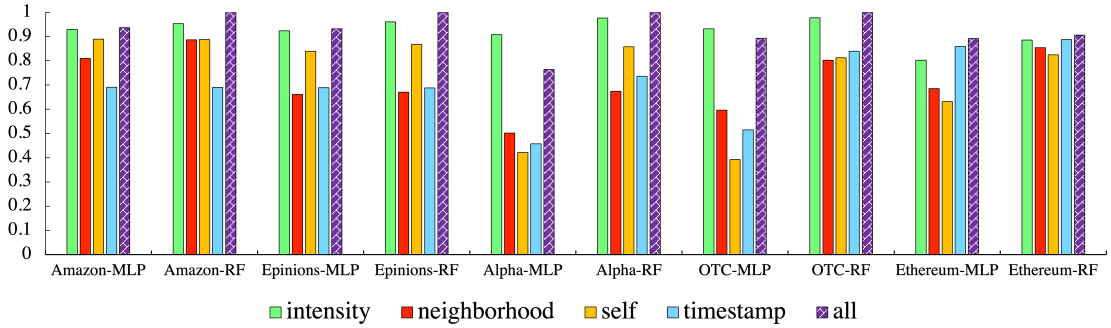

Figure 2: Results of ablation study on the performance of different sets of TGBASE features in the static node classification task, where the reported metric is *AUC*.

where more powerful or complex models have lower performance, while TGBASE achieves a perfect results. It should be noted that labeled nodes in the Amazon dataset are extremely imbalanced which can explain the higher AUC values.

• *Epinions.* The results on Epinions dataset is presented in the seventh column of Table 4. Due to the huge size of Epinion network, node2vec and RiWalk did not converge in reasonable time. However, we can observe that TGBASE reaches significantly better performance compared to REV2 [10], which implies the effectiveness of the proposed feature set on this dataset.

Overall, TGBASE achieves the best perfomance across different networks, and GNN methods (in case they did not employ TGBASE for generating node initial features) mostly achieve lower performance on all datasets compared to other methods. There are two reasons that can explain the lower performance of GNN methods when using one-hot encoding as initial features. First, GNN methods considerably depend on the initial features of the nodes, however, the considered datasets do not provide any initial node features. Second, GNN methods performs end-to-end node classification where they optimize the loss function to achieve high accuracy. However, these datasets are highly unbalanced and high accuracy can be achieved even by incorrectly predicting all instances as negative. Thus, although the methods can achieve high accuracy, we can observe that other performance measures, including AUC, which are more proper for unbalanced datasets, indicate a low performance when initial features are not expressive enough.

4.2 Impact of Different Groups of TGBASE features.

To better evaluate the impact of different TGBASE's sets of features, we conducted node classification when only one set of TGBASE features are used. Particularly, we evaluate the node classification performance, when only intensity, neighborhood, self, or timestamp features are employed as the node representations. As shown in Fig. 2, the aggregation of all features results in the best overall performance. Moreover, comparing the classification performance of individual feature sets, we can observe that intensity features can mostly achieve the best results, which implies the importance of incorporating the edge-weight based attributes in the node representations. Note that the Bitcoin dataset is anonymized and we could not define all sets of features for nodes of Bitcoin network, therefore this dataset is dropped here.

4.3 TGBASE for Dynamic Node Classification.

This section elaborates on evaluating the performance of TGBASE for dynamic node classification task against strong baselines that are specifically designed for representation learning on temporal graphs.

Datasets. For the experiments, we consider *Reddit* and *Wikipedia* datasets consisting of timestamped interactions of users on these social networks with evolving node labels corresponding to their reputation in the network. The goal of the classification task is to predict whether the state of a user will change due to an interaction. Specifically, for normal users that are not banned from posting sub-Reddits or editing Wikipedia pages, their label is always '0', while the label of a banned user changes to '1' after its final interaction.

• *Reddit.* This dataset consists of a bipartite graph including $1,000$ most active posts made by $10,000$ most active users on sub-Reddits during a period of one month [12]. The interactions among users and sub-Reddits are associated with text attributes representing the text information of the posts.

• *Wikipedia.* The dataset consists of a bipartite graph representing the edits made by users on Wikipedia pages. The $1,000$ most edited pages together with the users who made a minimum of 5 edits are considered as the nodes, and an edge demonstrates a user editing a page [12]. The interactions are associated with text attributes related to the page edits. Similar to the Reddit dataset, the edge weights of the Wikipedia network also consist of vectors

Table 5: Comparing the performance of TGBASE in terms of AUC with SOTA baselines in *dynamic* node classification task. The **first**, **second**, and **third** best performing method are colored correspondingly.

Method	Wikipedia	Reddit
JODIE [2019]	0.862 (±0.004)	0.675 (±0.006)
DyRep [2019]	0.837 (±0.007)	**0.691** (±0.005)
TGAT [2020]	0.501 (±0.001)	0.502 (±0.001)
TGN [2020]	**0.881** (±0.001)	0.668 (±0.008)
TGBase + RF	**0.874** (±0.010)	**0.713** (±0.007)
TGBase + MLP	**0.882** (±0.004)	**0.730** (±0.001)

of attributes; thus, the intensity features of TGBASE are defined for each attributes of the edge weight vectors. Node labels and interactions are timestamped, and nodes are labeled based on their states which represent whether a user is banned from editing a page.

Baselines. For dynamic node classification task, we compared TGBASE with four SOTAs on representation learning on temporal networks including JODIE [12], DyRep [25], TGAT [31], and TGN [23]. According to node interaction timestamps, we did a chronological train-validation-test split with 70%-15%-15% in line with the baseline methods [12, 25, 31, 23].

• *JODIE* [12] models the interactions of the users and items in domains such as social networks. JODIE utilizes RNNs to update the representations of the source and target nodes every time an interaction takes place.

• *DyRep* [25] intends to consider the topological evolution, as well as activities between the nodes to capture the dynamics of the interactions in the networks.

• *TGAT* [31] aims to aggregate temporal and topological features to recognize the node representations as function of time. It leverages GAT as its building blocks and develops functional time encoding

• *TGN* [25] is an efficient and generic framework that combines graph-based operators and memory modules for representation learning on temporal graphs.

Performance Evaluation. For Reddit and Wikipedia datasets, we predict the dynamic node labels, as related to the reputation ranking. Particularly, the downstream dynamic node classification task is used to predict whether the user is banned. The results are illustrated in Table 5.

Results. In Table 5, we can see that TGBASE obtains state-of-the-art results on both benchmark datasets. On Wikipedia, results are on par with TGN, whereas TGBASE significantly outperforms the baselines on Reddit dataset. We want to reemphasize that the features we extract require zero-learning or parameter adjustment. Yet, coupled with a shallow classifier, they are outperforming SOTA methods across different domains and settings.

5 Conclusions

In this paper, we proposed TGBASE, a simple yet powerful method for node classification in weighted temporal graphs. TGBASE encodes each node by extracting a small set of features based on the structural attributes of the node and its neighbors, as well as the intensity and timestamp attributes of the interactions among node pairs. Through extensive set of experiments we show that our simple shallow model outperforms more complex models which are currently the state-of-the-art in static and dynamic node classification task. Moreover, our method is more general compared to these methods, as they are defined per specific datasets or a class of datasets. TGBASE is therefore *generic* and generates efficient node representations for all the available benchmark datasets that we know of. Given its low time and model complexity, our proposed TGBASE is perfectly suited as a baseline when designing models for node classification in temporal graphs.

Acknowledgment

This research is partially funded from the Canada CIFAR AI Chairs Program.

References

[1] C. Akcora. Bitcoinheist: Topological data analysis for ransomware prediction on the bitcoin blockchain. In *Proceedings of the 2020 International Joint Conference on Artificial Intelligence (IJCAI)*, 2020.

[2] I. Chami, S. Abu-El-Haija, B. Perozzi, C. Ré, and K. Murphy. Machine learning on graphs: A model and comprehensive taxonomy. *arXiv preprint arXiv:2005.03675*, 2020.

[3] W. Chen, X. Guo, Z. Chen, Z. Zheng, and Y. Lu. Phishing scam detection on ethereum: Towards financial security for blockchain ecosystem. In *International Joint Conferences on Artificial Intelligence Organization*, pages 4506–4512, 2020.

[4] S. Farrugia, J. Ellul, and G. Azzopardi. Detection of illicit accounts over the ethereum blockchain. *Expert Systems with Applications*, 150:113318, 2020.

[5] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[6] W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035, 2017.

[7] S. M. Kazemi, R. Goel, K. Jain, I. Kobyzev, A. Sethi, P. Forsyth, and P. Poupart. Representation learning

for dynamic graphs: A survey. *Journal of Machine Learning Research*, 21(70):1–73, 2020.

[8] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[9] W. Kudo, M. Nishiguchi, and F. Toriumi. Gcnext: Graph convolutional network with expanded balance theory for fraudulent user detection. *Social Network Analysis and Mining*, 10(1):1–12, 2020.

[10] S. Kumar, B. Hooi, D. Makhija, M. Kumar, C. Faloutsos, and V. Subrahmanian. Rev2: Fraudulent user prediction in rating platforms. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 333–341, 2018.

[11] S. Kumar and N. Shah. False information on web and social media: A survey. *arXiv preprint arXiv:1804.08559*, 2018.

[12] S. Kumar, X. Zhang, and J. Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1269–1278, 2019.

[13] P. Li, Y. Wang, H. Wang, and J. Leskovec. Distance encoding: Design provably more powerful neural networks for graph representation learning. *arXiv preprint arXiv:2009.00142*, 2020.

[14] D. Lin, J. Wu, Q. Yuan, and Z. Zheng. Modeling and understanding ethereum transaction records via a complex network approach. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2020.

[15] X. Ma, G. Qin, Z. Qiu, M. Zheng, and Z. Wang. Riwalk: Fast structural node embedding via role identification. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 478–487. IEEE, 2019.

[16] P. Massa and P. Avesani. Trust-aware recommender systems. In *Proceedings of the 2007 ACM conference on Recommender systems*, pages 17–24, 2007.

[17] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26:3111–3119, 2013.

[18] L. Nan and D. Tao. Bitcoin mixing detection using deep autoencoder. In *2018 IEEE Third international conference on data science in cyberspace (DSC)*, pages 280–287. IEEE, 2018.

[19] P. Nerurkar, S. Bhirud, D. Patel, R. Ludinard, Y. Busnel, and S. Kumari. Supervised learning model for identifying illegal activities in bitcoin. *Applied Intelligence*, pages 1–20, 2020.

[20] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[21] T. Pourhabibi, K.-L. Ong, B. H. Kam, and Y. L. Boo. Fraud detection: A systematic literature review of graph-based anomaly detection approaches. *Decision Support Systems*, 133:113303, 2020.

[22] F. Poursafaei, R. Rabbany, and Z. Zilic. Sigtran: Signature vectors for detecting illicit activities in blockchain transaction networks.

[23] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*, 2020.

[24] S. Sayadi, S. B. Rejeb, and Z. Choukair. Anomaly detection model over blockchain electronic transactions. In *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pages 895–900. IEEE, 2019.

[25] R. Trivedi, M. Farajtabar, P. Biswal, and H. Zha. Dyrep: Learning representations over dynamic graphs. In *International conference on learning representations*, 2019.

[26] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[27] Y. Wang, Y.-Y. Chang, Y. Liu, J. Leskovec, and P. Li. Inductive representation learning in temporal networks via causal anonymous walks. *arXiv preprint arXiv:2101.05974*, 2021.

[28] M. Weber, G. Domeniconi, J. Chen, D. K. I. Weidele, C. Bellei, T. Robinson, and C. E. Leiserson. Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. *arXiv preprint arXiv:1908.02591*, 2019.

[29] J. Wu, D. Lin, Z. Zheng, and Q. Yuan. T-edge: temporal weighted multidigraph embedding for ethereum transaction network analysis. *arXiv preprint arXiv:1905.08038*, 2019.

[30] J. Wu, Q. Yuan, D. Lin, W. You, W. Chen, C. Chen, and Z. Zheng. Who are the phishers? phishing scam detection on ethereum via network embedding. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2020.

[31] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan. Inductive representation learning on temporal graphs. *arXiv preprint arXiv:2002.07962*, 2020.

[32] Z. Yuan, Q. Yuan, and J. Wu. Phishing detection on ethereum via learning representation of transaction subgraphs. In *International Conference on Blockchain and Trustworthy Systems*, pages 178–191. Springer, 2020.