TBoost: Gradient Boosting Temporal Graph Neural Networks

Pritam Kumar Nath
AI Garage
Mastercard
pritamkumar.nath@mastercard.com

Govind Waghmare
AI Garage
Mastercard
govind.waghmare@mastercard.com

Nancy Agrawal
AI Garage
Mastercard
nancy.agrawal@mastercard.com

Nitish Srivasatava
AI Garage
Mastercard
nitish.srivasatava@mastercard.com

Siddhartha Asthana
AI Garage
Mastercard
siddhartha.asthana@mastercard.com

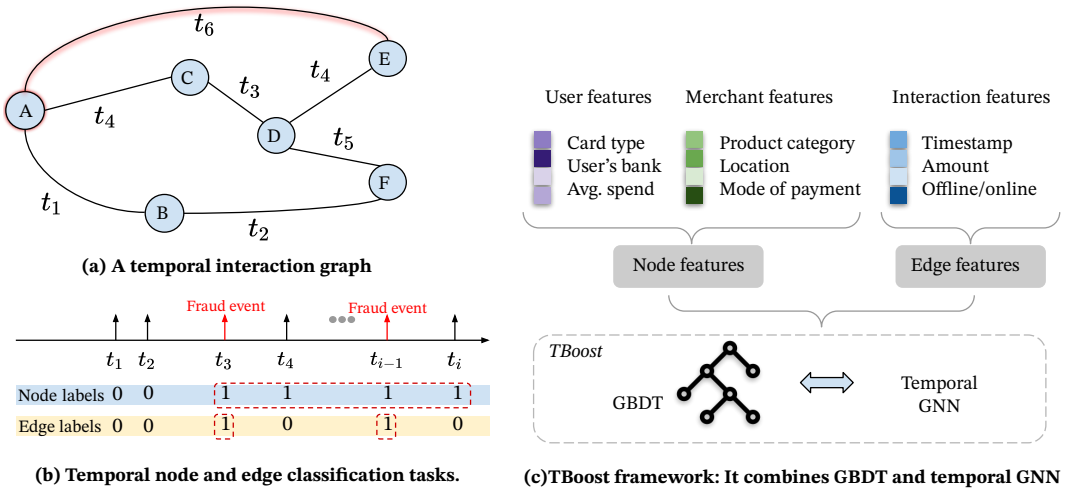
Abstract

Fraud prediction, compromised account detection, and attrition signaling are vital problems in the financial domain. Generally, these tasks are temporal classification problems as labels exhibit temporal dependence. The labels of these tasks change with time. Each financial transaction contains heterogeneous data like account number, merchant, amount, decline status, etc. A financial dataset contains chronological transactions. This data possesses three distinct characteristics: heterogeneity, relational structure, and temporal nature. Previous efforts fall short of modeling all these characteristics in a unified way. Gradient-boosted decision trees (GBDTs) are used to tackle heterogeneity. Graph Neural Networks (GNNs) are employed to model relational information. Temporal GNNs account for temporal dependencies in the data. In this paper, we propose a novel unified framework, TBoost, which combines GBDTs and temporal GNNs to jointly model the heterogeneous, relational, and temporal characteristics of the data. It leverages both node and edge-level dynamics to solve temporal classification problems. To validate the effectiveness of TBoost, we conduct extensive experiments, demonstrating its superiority in handling the complexities of financial data.

1 Introduction

A typical financial transaction is an interaction between entities like the cardholder, merchant, cardholder's bank, and merchant's bank. Each interaction is represented using different heterogeneous attributes like timestamp, amount, location, type of purchase (online or offline), mode of payment, domestic vs cross-border transaction, etc. For a given cardholder, understanding the purchase pattern over a certain period is crucial for customer lifetime value (LTV), fraud prevention, gross spend forecasting, churn prediction, and targeted marketing. Undoubtedly, a unified framework capable of learning heterogeneous transactional data along with temporal evolution is needed to model these characteristics.

Temporal Graph Learning Workshop @ NeurIPS 2023, New Orleans.



(a) A temporal interaction graph

(b) Temporal node and edge classification tasks.

(c)TBoost framework: It combines GBDT and temporal GNN

Figure 1: (a) A temporal interaction graph with source node A and target node E at time t_6. The goal of the edge-classification task is to predict the label associated with this particular interaction. On the other hand, for the node-classification task, the objective is to predict the label of the source node A at time t_6. (b) In temporal node classification task the state of node gets fixed when a event occurs on the node while in edge classification is the classification of the event itself. (c) An interaction contains a static node feature (eg. user and merchant) and a time-dependent edge feature. TBoost framework combines two modules GBDT and temporal GNN to learn the dynamics of heterogeneous transactional data.

With the increase in digital banking systems, there is a boost in the number of transactions performed online. Many banks try to improve the customer payment experience by providing attractive solutions like contactless and card-on-file. Although these methods reduce the steps involved in the authentication and verification of the cardholder, they might be vulnerable to fraud attacks. Malicious parties steal the credentials to purchase assets leading to banking and e-commerce fraud. Therefore, a scalable fraud prevention system is required to identify and stop such fraud activities.

A simple rule-based system on transactional attributes to identify fraudulent patterns can be effective due to simplicity and scalability. But, as fraud patterns evolve, it might need thousands of rules to capture each specific fraud scenario, and many domain experts to design and update these rules. Classical machine learning (ML) models capture the fairly complex fraud patterns in the data. For example, GBDT, a gradient-boosted decision tree, can learn the relationship between heterogeneous attributes of each transaction. However, the problem with classical ML models is effective transaction data representation. It becomes difficult the incorporate past purchase patterns and interactions (contextual information) within each transaction row of tabular data. Temporal events do not occur in isolation in the network. For example, a cardholder's previous purchase history might be influencing the purchase in the current transaction. The independent and identically distributed (iid) assumption on each transaction discards the structural and temporal information of the network. These models also require data processing pipelines to filter and transform data into a tabular format. GNNs capture the relationships between different entities (node types) involved in the financial network. The purchase trends or cardholders' behavior can be modeled as a temporal GNN. While, temporal GNNs model the interactions between heterogeneous entities (cardholders, banks, and merchants), GBDTs properly capture the heterogeneous attributes of the transaction. So, a novel framework combining the advantages of both GBDT and temporal GNN could be the right choice to model complex temporal data with heterogeneous node attributes.

Few prior works [6, 1] explore the possibility of learning graphs with tabular data. These frameworks combine the GBDT's learning on tabular data with GNNs learning on structural data. Prior frameworks do not consider the temporal ordering in the graph data which is essential in the financial fraud problems. They train on the static node features leading to suboptimal results in temporal classification tasks, where the label is a function of time. These methods do not use time-dependent edge features during training. For example, in fraud detection, the attributes of the fraudulent transaction

are more important than the static cardholder or merchant attributes. These attributes can provide information about the nature of the fraud, such as the card present indicator, cross border indicator, the amount of money involved, and the location. The framework based on GBDT and GNNs should be capable of handling heterogeneous node features. It should encode structural, temporal, and contextual information about the network. It also should be inductive to infer unseen nodes as one can observe entities not part of the training dataset. An example of inductiveness is a new cardholder or new merchant in the financial network.

In this work, we propose a novel framework called **TBoost**, which combines boosting with any temporal GNN. It contains two modules GBDT and temporal GNN. GBDT models the property of the events by learning decision trees on the combination of node and edge features. For the model to differentiate between different timestamps, we extend the edge feature vector with a time-encoded vector. The temporal GNN module uses the prediction of the GBDT module. It captures the node-level temporal relational structure by temporal neighborhood aggregation. Hence, TBoost models the data at both the edge and the node level by considering the historical events on the node. A high-level overview of TBoost is shown in Figure 1. In summary, we make the following contributions:

- We propose a novel framework TBoost to combine GBDT and temporal GNN. The GBDT models the heterogeneous attributes of the interaction and temporal GNNs learn the structural, temporal, and context of the interaction.
- Compared to prior works, we combine both node and edge features to provide better temporal information to GBDT. It is validated on temporal classification tasks where GBDT consistently shows an uplift due to temporal dynamics.
- In TBoost, we show experimental evaluation on two well-known temporal GNNs (TGN and TGAT). The experimentation on eight datasets for temporal node and edge classification task highlights the effect of TBoost. We also study the role of the pretrained temporal GNNs on performance.

2 Preliminaries

2.1 Dynamic Graphs

The dynamic graph represents a sequence of events occurring at different timestamps. Let $\mathcal{V}$ be the vertex set and $u, v \in \mathcal{V}$ are node indices. Then, dynamic graph is denoted as $\mathcal{G} = (e_{u,v}(t_1), e_{u,v}(t_2), \dots)$. Each event describes the interaction between two nodes with indices u, v at time t and events as time ordered as $t_1 \leq t_2 \leq \dots$. The edge set is $\mathcal{E} = \{(u, v), e_{u,v}(t) \in \mathcal{G}, t \in \mathbb{R}^+\}$. For example, the user-merchant interaction could be represented using a dynamic graph. The edge $e_{u,v}(t)$ denotes purchase by user $u \in \mathcal{V}$ on a merchant node $v \in \mathcal{V}$ at time t. As there could be multiple edges between the same user and merchant, representing multiple item purchases, $\mathcal{E}$ might be a multiset. Let $\boldsymbol{x}_u$ denotes the node features and $\boldsymbol{x}_{u,v}(t)$ denotes the feature vector corresponding to the edge between nodes u and v at timestamp t.

2.2 Gradient Boosted Decision Trees

Gradient Boosted Decision Trees (GBDT) [2, 11, 14] is a popular model for tabular and iid datasets. It is adept at handling heterogeneous features. Input to the GBDT are m training examples $(\boldsymbol{x}, y)$ where $\boldsymbol{x}$ is the input feature vector and y is the target. At each iteration i, a new weak learner h_i is selected from some candidate function space $\mathcal{H}$ (typically decision trees). Training of a GBDT involves minimizing the loss function $\mathcal{L}^{gbdt}(y, h_i(\boldsymbol{x}))$ over the training dataset, where $h_i(\boldsymbol{x})$ is the value predicted by the ith learner. For classification tasks, $\mathcal{L}^{gbdt}$ takes the form of cross-entropy, and for regression tasks mean the squared loss is employed. The h_i is trained on the negative gradient of $\mathcal{L}^{gbdt}$ with respect to the previously trained model $f_{i-1}(\boldsymbol{x})$.

$$h_i = \underset{h \in \mathcal{H}}{\arg\min} \sum_{j=1}^{m} \left\| -\frac{\partial \mathcal{L}^{gbdt}(y_j, f_{i-1}(\boldsymbol{x}_j))}{\partial f_{i-1}(\boldsymbol{x}_j)} - h_i(\boldsymbol{x}_j) \right\|_2^2 \tag{1}$$

Table 1: Comparison of TBoost with other baselines. CatBoost+ is a vanilla CatBoost with temporal features.

Method	Temporal	Inductive	Relational	Edge-Features
CatBoost[11]	✗	✓	✗	✓
CatBoost+	✓	✓	✗	✓
BGNN [6]	✗	✓	✓	✗
EBBS [1]	✗	✗	✓	✗
TBoost	✓	✓	✓	✓

The output of the current model $f_i(\boldsymbol{x})$ is then updated by adding a fraction λ of the output of the new weak learner $h_i(\boldsymbol{x})$, as follows:

$$f_i(\boldsymbol{x}) = f_{i-1}(\boldsymbol{x}) + \lambda h_i(\boldsymbol{x}) \tag{2}$$

λ is the learning rate which controls the step size of the update and helps to prevent overfitting. The process is repeated until the desired number of weak learners is reached. The final model is the sum of all the weak learners weighted by their learning rate.

2.3 Graph Neural Networks

Graph Neural Networks (GNNs) use graph topology to learn node representations. They aggregate information from neighboring nodes through message-passing.

Numerous message-passing approaches exist, such as averaging in GCN [7], pool and LSTM-based aggregators in GraphSage [5], attention-based aggregation in GAT [17], and node position-based aggregation in GIN [20]. Temporal dependency in GNNs can be modeled in two ways: Discrete-time and Continuous-time.

Discrete-time graph algorithms create representations for each temporal graph snapshot independently and combine them based on heuristics, resulting in the loss of crucial graph evolution information. For instance, DynGemm [4] uses an autoencoder to build representations incrementally, Dynamic-Triad [22] employs a triadic closure process, and EvolveGCN [10] and DySAT [13] use GNNs to combine node embeddings.

Continuous-time Graph considers each temporal event as a sequential edge formation in the graph evolution. For example, CTDNE [9] and CAW [18] are based on random walks. They learn node representations by aggregating the features of the nodes visited along the walk. TGAT [19] is a GNN-based method. It uses a self-attention mechanism, similar to GAT [17], along with functional time encoding to learn node representations. TGN [12] combines techniques from previous work and proposes a generic inductive framework. JODIE [8] and TigeCMN [21] focus on bipartite interactions in the network.

2.4 Previous methods: GNN + Boosting

The AdaGCN model [15] proposes a GNN architecture motivated by the AdaBoost iterations; however, this approach does not handle tabular data. BGNN [6] introduces a novel architecture that combines GBDT and GNN models to jointly leverage node features and graph structure, leading to significant performance gains on graph datasets with tabular features through iterative refinement.EBBS [1] introduces an algorithm that incorporates graph propagation and a unified bi-level boosting objective with a principled meta-loss function for reliable convergence.However, both EBBS and BGNN are limited to handling static graphs and are not suitable for tasks where the graph data changes over time. The lack of temporal adaptability restricts their relevance in scenarios with temporal dynamics. Moreover, the previous methods mainly apply boosting on the node features alone, which is suboptimal for temporal classification tasks. They lack an understanding of edge sequence formation. Some prior works [1] lack inductive property, which is the ability to evaluate the nodes not seen in the training. Inductivity is a serious concern for most of the practical financial networks as new users and merchants may get added to the system with time and retraining a large network is infeasible at the test time. In TBoost, we introduce a framework to combine temporal GNNs with the GBDTs. We

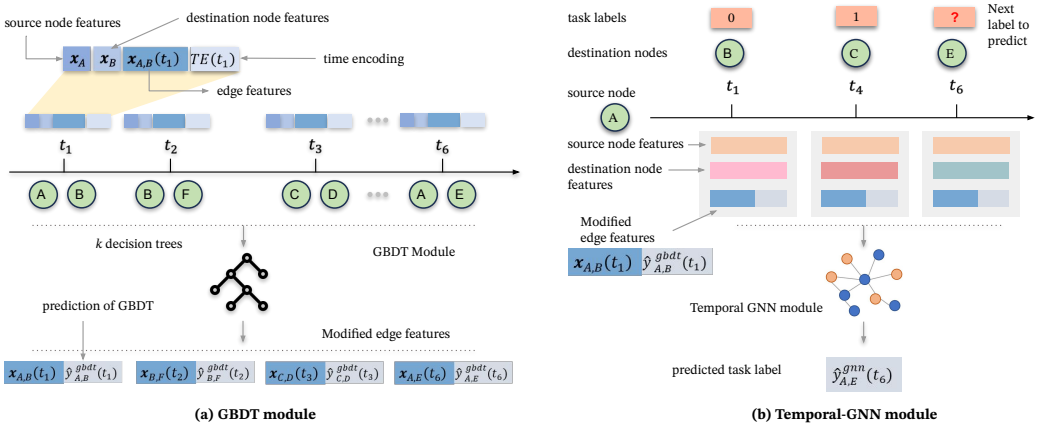


Figure 2: (a) Node features $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$ along with the edge feature $\boldsymbol{x}_{A,B}$ are concatenated with the Temporal Encoding $TE(t_1)$ to learn k decision trees. The output is the modified edge feature containing prediction $\hat{y}^{gbdt}_{A,B}(t_1)$ (b) The concatenated edge feature from the GBDT is passed through the temporal GNN along with the node features $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$ to predict task label $\hat{y}^{gnn}_{A,E}(t_6)$.

provide a simple way to add temporal information within the boosting algorithm itself. In TBoost, boosting is applied at the event level by combining the edge properties with the node properties. The temporal GNNs used in TBoost are inductive in nature. We compare the prior works in Table 1.

3 TBoost

TBoost framework consists of two main components: the GBDT module and the temporal GNN module. The GBDT module is responsible for modeling the specific event occurring at a particular timestamp. The temporal GNN models the importance of past events on the node. Both modules help in temporal classification tasks. TBoost boosts the temporal GNN by forcing the GBDT to learn from the mistakes of the GNN. The two components are trained end-to-end in an alternate fashion. In the context of GBDT , it's not possible to fine-tune existing trees because of their discrete structure, so instead, we iteratively enhance the model by adding new trees that approximate the GNN loss function. In the next few sections, we discuss the GBDT and temporal GNN modules and the strategy used for training TBoost. The architectural details of TBoost are shown in Figure 2.

3.1 GBDT module

Tabular data is structured data, with rows as observations and columns as features. Real-world graph datasets include tabular node and edge features. Temporal graphs contain timestamps for tracking event evolution. GBDTs handle high-dimensional and different-scaled data effectively. They learn hyperplane-like boundaries and converge faster compared to neural nets. GBDTs use static data in classification and regression tasks. They lack the temporal evolution aspect of data. To address this, we propose two ways to add temporal information to GBDTs: the temporal encoding function and the accommodation of edge features.

3.1.1 Temporal Encoders:

We explore the following temporal encoding (TE) methods to add temporal information into GBDTs. The impact of each encoding is shown in the experimental section.

Timestamp as a feature: Here, time is added as a scalar feature in the GBDT as TE(t) $= t$. It is a simple approach to integrate temporal dependence in tabular data.

Inter-event Time: Absolute timestamps could have large values with smaller relative differences. To resolve this, inter-event time may be used to encode time with an optional projection layer as shown in [8]. Here, TE$(t) = \delta t.w$, with w as the learnable weight and δt is the time difference from the previous event or edge.

Functional Time Encoder: [19] introduced a learnable time encoding kernel based on harmonic analysis (Bochner's theorem).

$$\text{TE}(t) = \sqrt{\frac{1}{d}}[\cos(\omega_1 t), \sin(\omega_1 t), ..., \cos(\omega_d t), \sin(\omega_d t)]$$

where, ω_i is learnable frequency and d is the embedding dimension.

Recently, GraphMixer[3] empirically proves that using a functional encoder with fixed weights converges faster and boosts performance. The learning might become unstable due to exploding gradients. Bochner's kernel is a popular choice for machine-learning applications. Other choices of kernel include the Hawkes kernel, the exponential kernel, and the Gaussian kernel. Note that, due to the discrete structure of trees, it is difficult to optimize parameter weights for the time encoder in the GBDTs. We used fixed-weighted versions of the encoders in our experiments.

3.1.2 Edge Features in Temporal Classification:

In temporal classification tasks, the edge features $\boldsymbol{x}_{u,v}(t)$ carry information about the present event and are required to model the state of a node/edge. For example, in a financial network, a card might become compromised if fraud occurs on the card. To model the fraud event, we need the transaction attributes in addition to the properties of the card, such as the time-independent cardholder details (country, age, issuer ID, etc.). BGNN [6] and EBBS [1] model only node dynamics while TBoost models both node and edge dynamics. Formally, the edge features $\boldsymbol{x}_{u,v}(t)$ and node features $\boldsymbol{x}_u$ and $\boldsymbol{x}_v$, along with the timestamp t, are input to the GBDT module. First, the time is encoded using the temporal encoder function (TE) and aggregated with the features. The time encoder function is used to capture the temporal dynamics of the graph. This is important because temporal graph data is often sequential, and the order of events can be important for making predictions. Node, edge, and time features are combined into a single feature vector $\boldsymbol{x}_{u,v}^{gbdt}(t)$ that is used for training decision trees as shown in Figure 2(a). It is defined as follows:

$$\boldsymbol{x}_{u,v}^{gbdt}(t) = [\boldsymbol{x}_u, \boldsymbol{x}_v, \boldsymbol{x}_{u,v}(t), \text{TE}(t)] \tag{3}$$

3.2 Temporal GNN module

While GBDTs are suitable for tabular data, they struggle to represent the underlying network topology. GBDT requires additional preprocessing and manually engineered graph features to capture the relational structure. Contrary, GNNs can capture rich structural information in network data. GNNs incorporate neighborhood information and node features for predictions integrating the graph structure. Trends and seasonality are common in real-world graph data. The omission of the event times might hurt the learning of these temporal patterns. TBoost facilitates temporal classification tasks by incorporating a temporal GNN module that is compatible with any continuous graph models. This includes a range of techniques such as RNN-based approaches like JODIE [8], temporal point process approaches like Dyrep [16], self-attention-based methods like TGAT [19] and TGN [12], among others. In our experiments, we focus on TGAT and TGN due to their superior performance in temporal classification tasks as documented in the existing literature.

3.3 Training and Loss Propagation

In this section, we present TBoost's training strategy, a framework designed for node/edge classification in temporal graphs. The training process involves an alternate optimization of the GBDT module and the temporal GNN module for N iterations. The algorithm of TBoost has been summarized in supplementary section 1.1.

Dataset Partitioning: The dataset is partitioned into training, validation, and test sets. It is done by dividing the entire time-space in the split ratio of 70%, 15%, and 15% respectively, inline with the previous baselines [19, 12]. This is the inductive setting we would be following in the experiments.

GBDT Model Construction: In training, the first iteration involves constructing a GBDT model, denoted as f_1, with k decision trees. The model is built by minimizing the loss function $\mathcal{L}^{gbdt}$, which depends on the specific task at hand. The input to the GBDT model is the modified feature $\boldsymbol{x}_{u,v}^{gbdt}(t)$, as shown in Equation 3.

Modified Edge Features for GNN: The predictions from the GBDT model are concatenated with the original edge features to form the modified edge features as $\bar{\boldsymbol{x}}_{u,v}(t) = \boldsymbol{x}_{u,v}^{gbdt}(t) \parallel \hat{y}_{u,v}^{gbdt}(t)$. These features are passed as input to the temporal GNN module. Note that, only the edge features undergo modifications, while the node features and the edge connectivity of the graph remain the same. The GBDT module focuses on identifying the label of the present edge or event, disregarding the impact of previous events on the same node. On the other hand, the temporal GNN module captures the sequence of past events taking place on the nodes.

Training the GNN Model: For a temporal GNN with learnable parameters g_θ, the input is again a concatenation of modified edge features with node features as $\boldsymbol{x}_{u,v}^{gnn}(t) = \boldsymbol{x}_u \parallel \boldsymbol{x}_v \parallel \bar{\boldsymbol{x}}_{u,v}(t)$. Along with $\boldsymbol{x}_{u,v}^{gnn}(t)$, we pass temporal graph $\mathcal{G}$ to the training function for neighborhood information. The training process involves minimizing the loss of function $\mathcal{L}^{gnn}(t)$ using l steps of gradient descent :

$$\theta, -\eta \frac{\partial \mathcal{L}_{gnn}}{\partial \boldsymbol{x}_{u,v}^{gnn}(t)} \overset{l}{\leftarrow} \arg\min_{\theta} \mathcal{L}^{gnn}(g_\theta(\mathcal{G}, \boldsymbol{x}_{u,v}^{gnn}(t)), y(t)) \tag{4}$$

Target Generation for GBDT: Starting from the next iteration, the gradient of the loss function $\mathcal{L}^{gnn}$ with respect to the input features $x_{u,v}^{gnn}(t)$ is used as the target for the GBDT f_2. This approach is inspired by the BGNN framework [6], where f_2 aims to fit the direction that improves GNN predictions based on the initial GBDT predictions $f_1(\boldsymbol{x}_{u,v}^{gbdt}(t))$. The target for the GBDT module, denoted as $y_{u,v}^{gbdt}(t)$, is defined as:

$$y_{u,v}^{gbdt}(t) = \begin{cases} y_{u,v}(t), & \text{if } iter = 1 \\ -\eta \frac{\partial \mathcal{L}_{gnn}(g_\theta(\mathcal{G}, \boldsymbol{x}_{u,v}^{gnn}(t)), y(t))}{\partial \boldsymbol{x}_{u,v}^{gnn}(t)}, & \text{otherwise} \end{cases} \tag{5}$$

Here, η represents the learning rate.

Sum Model Combination: Once f_2 is trained, the predictions $f = f_1 + f_2$ are combined to form the sum model f. The sum model is the output of the GBDT module. In the first iteration, it simply returns the learned GBDT model, while in subsequent iterations, it returns the sum of all the models learned so far. The sum model is then used to predict the target, which is combined with the features to form $\bar{\boldsymbol{x}}_{u,v}(t) = \boldsymbol{x}_{u,v}^{gbdt}(t) \parallel \hat{y}_{u,v}^{gbdt}(t)$. It is input to the temporal GNN module as shown in Figure2(b).

Model Output: In total, TBoost is trained for N iterations, producing a GBDT model $f : \boldsymbol{x}_{u,v}^{gbdt}(t) \rightarrow \hat{y}_{u,v}^{gbdt}(t)$ and a GNN model $g_\theta : (\mathcal{G}, \boldsymbol{x}_{u,v}^{gnn}(t)) \rightarrow \hat{y}_{u,v}^{gnn}(t)$. These models are utilized for downstream tasks like temporal node and edge classification. In the node classification, the node embedding is passed to a Multilayer Perceptron (MLP). In the edge-level classification task, the embeddings of the two associated nodes are concatenated and passed through the MLP. In both cases, the cross-entropy loss is optimized. Time complexity of each pass of the algorithm is same as GBDT and GNN combined but the number of epochs needed to reach best performance is much lower because of the confidence boost by GBDT.

4 Experiments

In this section, we discuss our experiments and results, with further details on baselines and datasets provided in Supplementary Section 1.2. We exclude previous boosting-GNN methods such as [[6] and [1] from our experiments, as adapting them to our current settings is non-trivial due to their static, non-inductive nature.

4.1 Inductive Temporal Node Classification

As temporal graph $\mathcal{G}$ is continuously evolving, certain nodes get added to the network while features of existing nodes are updated. For example, in the financial transactions system, new cardholders and merchants join the network. An upgrade in the credit card will update the associated card's node features. When chargeback or fraud occurs on the card, the label associated with the card changes from normal to risky. Given an interaction $e_{u,v}(t)$ of the card u with merchant v, with edge features $\boldsymbol{x}_{u,v}(t)$, we define the node label $y_{u,v}(t)$ depicting the occurrence of fraud. If fraud is observed on the card, then for every subsequent transaction node label is $y_{u,v}(t) = 1$ otherwise, it is set to $y_{u,v}(t) = 0$

Table 2: Results on inductive temporal node classification using ROC-AUC. The best numbers are shown in bold. The next best numbers are shown in the underline.

Model Type	Model	Reddit	Wikipedia	MOOC	Payments	BankSim	Average
Boosting method	CatBoost	56.62	59.30	59.73	52.70	53.00	56.27
	CatBoost+	56.59	65.75	60.26	56.10	52.90	62.86
	XGBoost	53.34	55.50	59.82	52.20	52.20	54.61
	XGboost+	57.3	64.91	60.10	54.60	52.25	64.55
Temporal GNN	Jodie	61.83	84.84	63.00	58.20	65.89	68.75
	Dyrep	62.91	84.59	58.69	58.39	54.50	63.81
	TGAT	65.60	83.69	58.93	56.14	77.60	68.40
	TGN	<u>68.90</u>	87.11	<u>65.70</u>	<u>63.86</u>	<u>87.30</u>	<u>74.57</u>
GNN+Boosting	TGAT+	66.03	<u>87.50</u>	59.32	58.40	78.10	69.87
	TGN+	**69.30**	**88.30**	**66.68**	**66.80**	**88.10**	**75.84**

as shown in Figure 1. Similarly, merchants involved in fraudulent activities could be labeled. As new cards and merchants will not be part of the training set, inductive modeling is essential to infer these unseen entities. In node classification, we try to predict the label $y_{u,v}(t)$ for a given source node u. GBDTs underperform in node classification due to their inability to model node interactions. Combining GNNs with GBDTs in TBoost improves performance by jointly modeling event and node sequences. Three sections in Table 2: Boosting Methods (CatBoost, XGBoost), Temporal GNNs (JODIE, Dyrep, TGAT, TGN), and TBoost Framework. We extend GBDTs with temporal info (CatBoost+, XGBoost+) and experimented with encoding techniques (details in Section 1.5 of Supplementary). JODIE excels in learning bipartite graphs. TGN, with node memory, outperforms temporal GNNs. TBoost + TGN improves further by modeling heterogeneous features.

4.2 Inductive Temporal Edge Classification

In edge classification, given an edge $e_{u,v}(t)$, we predict its label $y_{u,v}(t)$. For example, fraud probability prediction for each transaction. Note that, edge classification is different than the node classification problem as it models edge-level events (likelihood of fraud or chargeback) while the other one models the node-level events (card-level risk). In edge classification, one needs to observe the history of both the source and target node involved (card and merchant) like how past transactions of the card are influencing the current purchase. In inductive settings, either the source or target nodes are newly seen in the network. It could be a new card or new merchant or both. The last case where both card and merchant are new is typically observed in money laundering. In edge classification (Table 3), GBDTs perform well on Luxury and Office, while GNNs excel in Fashion. GBDTs outperform GNNs when heterogeneous row features are crucial, while GNNs shine when current row features depend on past interactions. TBoost with GBDT and temporal GNN consistently achieves near-optimal results, combining feature interpretability from GBDTs and transfer learning from GNNs. Detailed analysis of how each module in TBoost performs is attached in Section 1.5 of supplementary material.

5 Conclusion

In this work, we discussed the limitation of prior frameworks in integrating boosting with temporal-GNNs. We overcome these shortcomings by introducing temporal encoder and edge features in the GBDT module of TBoost. The temporal GNN module can incorporate any continuous time GNN. TBoost is trained end-to-end for jointly modeling node-level and edge-level dynamics. Through extensive experimentation on 8 datasets with temporal node and edge classification, we show the merits of TBoost.

Table 3: Results on three Amazon datasets for inductive temporal edge classification using F1 score. The best numbers are shown in bold. The next best numbers are shown in the underline.

Model Type	Model	Fashion	Luxury	Office
Boosting method	CatBoost	71.28	54.1	73.02
	CatBoost+	71.60	54.28	**73.21**
	XGboost	72.96	**54.32**	72.60
	XGboost+	73.36	**54.32**	72.71
Temporal GNN	Jodie	77.98	53.66	72.40
	Dyrep	<u>78.30</u>	53.54	72.36
	TGAT	78.15	53.53	72.10
	TGN	78.20	53.53	72.50
GNN + Boosting	TGAT+	**79.6**	54.10	72.66
	TGN+	78.25	<u>54.30</u>	<u>72.81</u>

References

[1] Jiuhai Chen et al. "Does your graph need a confidence boost? Convergent boosted smoothing on graphs with tabular node features". In: *International Conference on Learning Representations*. 2022.

[2] Tianqi Chen and Carlos Guestrin. "Xgboost: A scalable tree boosting system". In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016.

[3] Weilin Cong et al. "Do We Really Need Complicated Model Architectures For Temporal Networks?" In: *The Eleventh International Conference on Learning Representations*. 2023.

[4] Palash Goyal et al. "DynGEM: Deep Embedding Method for Dynamic Graphs". In: *arXiv preprint arXiv:1805.11273* (2018).

[5] Will Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs". In: *Advances in neural information processing systems* (2017).

[6] Sergei Ivanov and Liudmila Prokhorenkova. "Boost then Convolve: Gradient Boosting Meets Graph Neural Networks". In: *International Conference on Learning Representations*. 2021.

[7] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *International Conference on Learning Representations*. 2017.

[8] Srijan Kumar, Xikun Zhang, and Jure Leskovec. "Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks". In: *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019.

[9] Giang Hoang Nguyen et al. "Continuous-Time Dynamic Network Embeddings". In: *Companion Proceedings of the The Web Conference 2018*. 2018.

[10] Aldo Pareja et al. "Evolvegcn: Evolving graph convolutional networks for dynamic graphs". In: *Proceedings of the AAAI conference on artificial intelligence*. 2020.

[11] Liudmila Prokhorenkova et al. "CatBoost: unbiased boosting with categorical features". In: *Advances in neural information processing systems* (2018).

[12] Emanuele Rossi et al. "Temporal Graph Networks for Deep Learning on Dynamic Graphs". In: *ICML 2020 Workshop on Graph Representation Learning*. 2020.

[13] Aravind Sankar et al. "DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks". In: *WSDM '20: The Thirteenth ACM International Conference on Web Search and Data Mining, Houston, TX, USA, February 3-7, 2020*. 2020.

[14] Robert P. Sheridan, Andy Liaw, and Matthew Tudor. *Light Gradient Boosting Machine as a Regression Method for Quantitative Structure-Activity Relationships*. 2021.

[15] Ke Sun, Zhanxing Zhu, and Zhouchen Lin. "AdaGCN: Adaboosting Graph Convolutional Networks into Deep Models". In: *International Conference on Learning Representations*. 2020.

[16] Rakshit Trivedi et al. "DyRep: Learning Representations over Dynamic Graphs". In: *International Conference on Learning Representations*. 2019.

[17] Petar Veličković et al. “Graph Attention Networks”. In: *International Conference on Learning Representations*. 2018.

[18] Yanbang Wang et al. “Inductive Representation Learning in Temporal Networks via Causal Anonymous Walks”. In: *International Conference on Learning Representations*. 2021.

[19] Da Xu et al. “Inductive representation learning on temporal graphs”. In: (2020).

[20] Jiaxuan You, Rex Ying, and Jure Leskovec. “Position-aware graph neural networks”. In: *International conference on machine learning*. Proceedings of Machine Learning Research. 2019.

[21] Zhen Zhang et al. “Learning Temporal Interaction Graph Embedding via Coupled Memory Networks”. In: *Proceedings of The Web Conference 2020*. 2020.

[22] Le-kui Zhou et al. “Dynamic Network Embedding by Modeling Triadic Closure Process.” In: *Association for the Advancement of Artificial Intelligence*. 2018.