Confidence First: Reliability-Driven Temporal Graph Neural Networks

Jayadratha Gayen
Machine Learning Lab, IIIT Hyderabad
Hyderabad, India
jayadratha.gayen@research.iiit.ac.in

Himanshu Pal
Machine Learning Lab, IIIT Hyderabad
Hyderabad, India
himanshu.pal@research.iiit.ac.in

Naresh Manwani
Machine Learning Lab, IIIT Hyderabad
Hyderabad, India
naresh.manwani@iiit.ac.in

Charu Sharma
Machine Learning Lab, IIIT Hyderabad
Hyderabad, India
charu.sharma@iiit.ac.in

Abstract

Many real-world systems can be modeled as dynamic graphs, where nodes and edges evolve over time, requiring specialized models to capture their evolving dynamics in risk-sensitive applications effectively. Graph neural networks (GNNs) for temporal graphs are one such category of specialized models. For the first time, our approach integrates a reject option strategy within the framework of GNNs for continuous-time dynamic graphs (CTDGs). This allows the model to strategically abstain from making predictions when the uncertainty is high and confidence is low, thus minimizing the risk of critical misclassification and enhancing the results and reliability. We propose a coverage-based abstention prediction model to implement the reject option that maximizes prediction within a specified coverage. It improves the prediction score for link prediction and node classification tasks. Temporal GNNs deal with extremely skewed datasets for the next state prediction or node classification task. In the case of class imbalance, our method can be further tuned to provide a higher weight to the minority class. Exhaustive experiments are presented on four datasets for dynamic link prediction and two datasets for dynamic node classification tasks. This demonstrates the effectiveness of our approach in improving the reliability and area under the curve (AUC)/average precision (AP) scores for predictions in dynamic graph scenarios. The results highlight our model's ability to efficiently handle the trade-offs between prediction confidence and coverage, making it a dependable solution for applications requiring high precision in dynamic and uncertain environments. Our code is available at https://github.com/Jayadratha/Cover_DyG

CCS Concepts: • **Theory of computation** → Dynamic graph algorithms; • **Computing methodologies** → *Uncertainty quantification*; Semi-supervised learning settings; *Temporal reasoning*; **Machine learning**; • **Information systems**;

Keywords: Temporal Graph Learning, Graph Neural Networks, Reject Option Classification, Link Prediction, Node Classification

ACM Reference Format:
Jayadratha Gayen, Himanshu Pal, Naresh Manwani, and Charu Sharma. 2025. Confidence First: Reliability-Driven Temporal Graph Neural Networks. In *Proceedings of Temporal Graph Learning Workshop, SIGKDD International Conference on Knowledge Discovery and Data Mining 2025 (TGL Workshop, KDD 2025).* ACM, New York, NY, USA, 14 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

In the modern era, numerous systems are modeled as dynamic graphs where nodes and edges evolve over time. These systems include social and interaction networks [22], traffic networks [1] trade networks [29], biological networks[2] and transaction networks[24] among others. Particularly in risk-sensitive applications such as fraud detection[30], fake news, polarization identification[6], financial transactions[24], blockchain security [3], epidemic modeling[38], anomaly detection[31] and disease prediction[3], the stakes for accurate and reliable predictions are exceptionally high.

Traditional graph neural networks (GNNs) and their extensions to dynamic graphs have shown promise in capturing the complex interactions and evolving structures within these networks. However, they often fall short where the cost of misclassification is significant and abstaining from making a prediction could mitigate risk. The additional challenge of highly skewed temporal[1] datasets for node classification

[1]"temporal" and "dynamic" will be used interchangeably.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
TGL Workshop, KDD 2025, Toronto, Canada
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

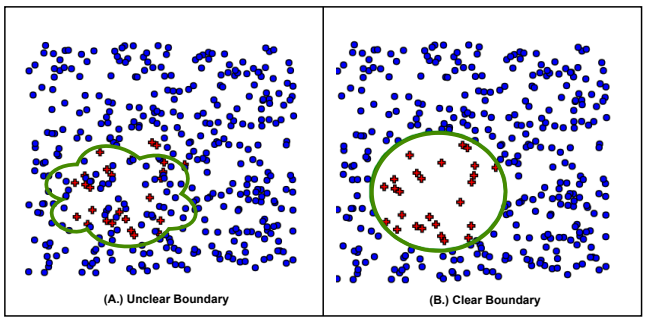

Figure 1. Decision Boundary: (A) is not clear and smooth; (B) becomes clearer after abstaining confusing/noisy samples. The examples to be rejected are decided by the objective function, which consists of the losses over unrejected samples and a fixed cost for rejected samples. The minimum value of the objective will be achieved only with a proper choice of rejected examples.

(such as Wikipedia, Reddit datasets[22]) makes it harder for the models to predict correctly. Recent works on temporal GNNs[7, 32, 41, 42] solve for node classification and link prediction tasks.

Real-world classification tasks, such as anomaly detection or credit card fraud prediction in dynamic networks, often involve overlapping class distributions and class imbalance, leading to prediction uncertainty. Figure 1(A) illustrates such a scenario. The minority class (red crosses) is infrequent and significantly overlaps with the majority class (blue circles) in a specific region. A classifier attempting to perfectly separate these classes might require a complex decision boundary (conceptualized by the irregular green line), potentially leading to overfitting or unreliable predictions for samples near this boundary.

Figure 1(B) conceptually motivates the benefit of abstention. By identifying samples with high prediction uncertainty (typically those within the overlap region), the model can choose to abstain from making a prediction for them. This allows the model to focus on making high-confidence predictions for the remaining samples, which might now be effectively separated by a simpler, more robust decision boundary (conceptualized by the circular green line). This improves both interpretability and trustworthiness, especially in settings where wrong decisions are costly.

Classifiers with abstention are well explored in the machine learning community[4, 12, 18, 23]. Such models allow a classifier to withhold a prediction when the uncertainty associated with the decision is high. This approach has not been widely investigated in the context of dynamic graphs, where the temporal evolution of the data adds a layer of complexity to uncertainty management.

Motivated by the critical need for precision in predictions, especially in environments where trade-offs between false positives and false negatives carry substantial consequences, we explore the integration of the abstention option with the continuous time dynamic graphs (CTDGs) learning method. The open problem, therefore, is to incorporate a mechanism that allows CTDGs to abstain from making predictions under high uncertainty, thus reducing the risk of misclassifications in sensitive applications. To address this, we propose a novel framework that integrates the reject option into CTDGs for both link prediction and node classification tasks. This framework aims to enhance model reliability by minimizing the risk associated with uncertain predictions.

We solve this challenge by extending the methodologies of temporal GNNs to incorporate a reject option classification. Our approach involves designing a specialized neural network architecture that dynamically adjusts its confidence threshold based on the evolving graph structure and node interactions. This method allows the model to strategically abstain from making predictions when uncertainty is high, leveraging a coverage-based model to optimize the trade-off between accurate prediction and the cost of abstention.

Our main goal is uncertainty management using the reject option. While analyzing the data, we find that extreme class imbalance is often present in node classification tasks for CTDG datasets. As uncertainty amplifies due to class imbalance and noisy boundary between the classes, our approach tries to handle both problems together. Previous works have struggled to effectively manage the disproportion between classes, which is particularly problematic in risk-sensitive domains where minority classes may carry significant importance. Our approach handles extreme class imbalance by optimizing the model's training to better represent minority classes, thereby significantly improving performance in these challenging scenarios. This advancement is a pioneering step in dynamic GNNs research, offering a powerful tool for domains where class imbalance profoundly affects the utility and reliability of the model. Our main contributions are as follows:

1. **Uncertainty management with reject option:** We integrate a reject option strategy into the framework of CTDGs. This empowers the model to abstain from making predictions when it lacks sufficient confidence, addressing the critical need for risk mitigation in sensitive applications. To achieve this, we propose a coverage-based approach to incorporate the reject option, allowing for the customization of the model's behavior to optimize predictions within a set coverage limit.
2. **Managing extreme class imbalance:** Drawing from recent literature for class imbalance mitigation, we address this problem in dynamic graph node classification, an area largely unexplored for extreme imbalance.
3. **Comprehensive evaluation:** We conduct extensive experiments on various dynamic graph datasets for link prediction and node classification to demonstrate

the effectiveness of our approach. Our results highlight the ability of our models to effectively manage trade-offs between prediction confidence and coverage while significantly improving performance metrics.

2 Related Work

Graph Neural Networks. GNNs[33] are the go-to method for dealing with graph data. GNNs address the limitation of learning representations[13] for nodes and edges. GNNs employ various message passing schemes (GCN[21], GraphSAGE[14], GAT[39] etc.) to aggregate and update node embedding, enabling them for prediction tasks like node classification, link prediction, graph classification, and graph clustering. PyG[11] offers a flexible framework based on Pytorch[27] for GNNs that allows to build models for specific graph problems. However, challenges remain in efficiently handling the evolving nature of dynamic graphs, where nodes and edges can change over time.

Temporal Graphs. Dynamic graphs [19] are categorized into discrete-time (DTDGs)[26] and continuous-time (CTDGs)[22] settings. In this paper, we focus on CTDGs. CTDGs are addressed using node-based methods, such as DyRep[37], TGN[32], and TCL[40] utilize node information, such as temporal neighbors and previous histories of nodes, to create node embedding, and edge-based methods like GraphMixer[7] and CAWN[41] directly generate embeddings for the edge of interest. DyGFormer[43] proposed a transformer-based architecture that uses a neighbor co-occurrence encoding scheme along with a patching technique to effectively capture long-term temporal dependencies in dynamic graphs. Recent methods, such as DyGMamba[9] and FreeDyG[35], incorporate state-space models and frequency-domain analysis for better encoding of interactions. TGB[17] benchmarks popular temporal graph models treated the task as a ranking problem. DyGLib[40] based on PyG[11], provides a unified library for implementing various CTDG methods. We utilize DyGLib to set up experiments and integrate a rejection module.

Uncertainty Estimation & Classification With Rejection (CwR). CwR can be classified broadly into cost-based and coverage-based methods. Cost-based CwR method aligns well with cost-sensitive learning[10] where the cost of misclassification can be used to inform the decision to abstain from prediction. Variants of support vector machine (SVM) with reject option are presented in [34]. [4] introduces a novel approach to multi-class classification with rejection compatible with arbitrary loss functions, addressing the need for flexibility in adapting to different datasets. In coverage-based CwR, SelectiveNet[12] introduces a novel approach by integrating the reject option directly within the deep neural network architecture. [5, 25] provide calibrated losses for multi-class reject option classification. [4] proposes a general recipe to convert any multi-class loss function to accommodate the reject option, calibrated to loss l_{0d1}. They treat rejection as another class. Conformal prediction (CP) provides valid uncertainty estimates by constructing prediction sets that contain the true label with a predefined confidence level. CF-GNN[15] integrates CP with GNNs to generate uncertainty-aware predictions on graph data. In our task for CTDGs, we only handle binary classification, unlike [8]. So, there is little scope to apply this method in this setting.

3 Preliminaries

This section introduces the concepts and notations used throughout this paper. Our work focuses on extending GNN to temporal graphs, incorporating a classification with a rejection option for enhanced prediction in both link prediction and node classification tasks.

Deep Learning on Static Graphs. It focuses on learning representations for graphs that do not change over time. A static graph is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ represents the set of edges connecting these nodes. Each node $u \in \mathcal{V}$ and edge $(u, v) \in \mathcal{E}$ can have associated features $\mathbf{w}_u$ and $\mathbf{e}_{uv}$, respectively. GNNs use features and the structure of the graph to learn a function $f : \mathcal{V} \rightarrow \mathbb{R}^d$. It aggregates information from the local neighborhoods of nodes through a series of message-passing operations that map nodes to a d-dimensional embedding space.

Deep Learning on Dynamic Graphs. Dynamic graphs are a sequence of non-decreasing chronological interactions to accommodate changes over time. They represent evolving relationships within the graph. A dynamic graph is represented as a series of graphs $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$ at discrete time steps t, or as a continuous stream of interactions $(u, v, t) \in \mathcal{E}_t$, where $u, v \in \mathcal{V}_t$ are nodes, and t represents the time of interaction. The primary goal in deep learning on dynamic graphs is to learn a function $f : \mathcal{V}_t \rightarrow \mathbb{R}^d$ that captures not only the structural features but also its temporal dynamics, facilitating tasks such as temporal link prediction and dynamic node classification.

Classification with Rejection. Rejection classification introduces a decision framework in which a model, given an input $x \in \mathcal{X}$, can choose to abstain from making a prediction if it lacks confidence. Given a prediction model $f : \mathcal{X} \rightarrow \mathcal{Y}$, where $\mathcal{X}$ is the input space and $\mathcal{Y}$ is the output label space, and an abstention function $q : \mathcal{X} \rightarrow \{0, 1\}$. For each sample x, the model predicts an abstention score $a(x)$ where $a : \mathcal{X} \in [0, 1]$. We first sort the abstention scores for all the interactions and then select a threshold θ value that aligns with the desired coverage. For an input x, if the abstention score, $a(x) \leq \theta$ then $q(x) = 0$ (the model is confident) and the model predicts $f(x)$. In case $a(x) > \theta$ then $q(x) = 1$ and the model abstains the decision for input x. For a given

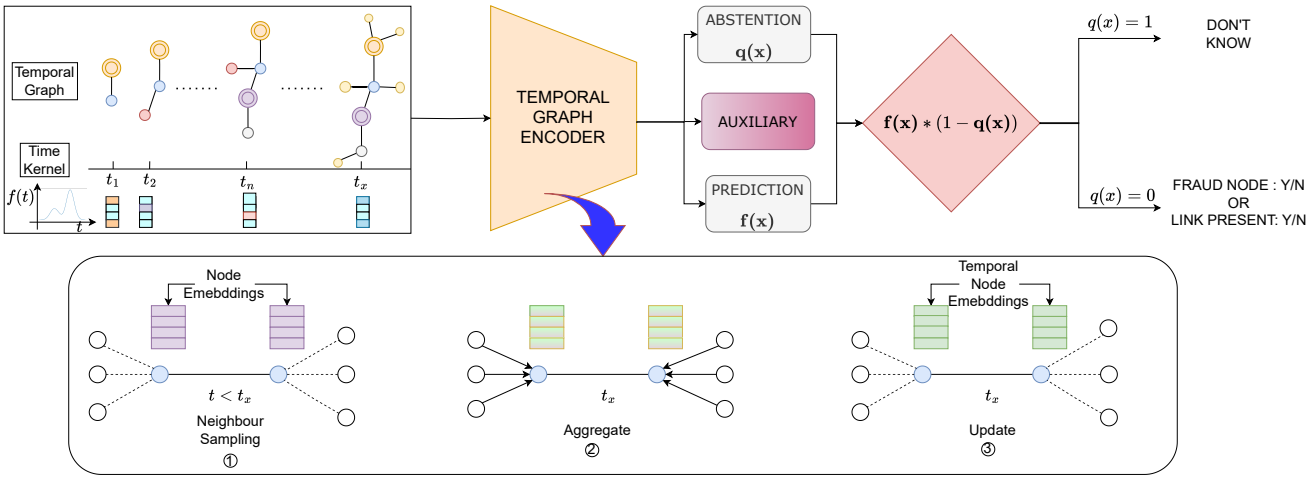

Figure 2. The model overview of classification with rejection for task-agnostic temporal graphs. Input to the model is a temporal graph with t timestamps, which is processed through an encoder that samples neighbours from all interactions from previous timestamps $t < t_x$ of these nodes, then aggregates and yields an updated temporal node embedding. These embeddings are passed to a *Prediction* function $f(x)$ to predict downstream tasks such as whether a link exists or not for link prediction or whether a user is blocked or not for node classification at time t. The output embeddings also pass through the *Abstention* function, predict a single score, and check if $a(x) > \theta$ threshold i.e. $q(x) = 1$. When it is true, the model abstains; otherwise, it proceeds to make a prediction. The *Auxiliary* head exposes all samples to the model, including samples with high abstention scores, only during training.

$\theta \in (0, 1)$, the objective is to maximize the probability of correct prediction for the subset of the inputs where $q(x) = 0$:

$$\max_{f,q} P(f(x) = y | q(x) = 0) = \max_{f,q} P(f(x) = y | a(x) \leq \theta) \tag{1}$$

Problem Formulation. Given a continuous-time temporal graph $\mathcal{G}_t$ with a tuple $(u, v, t) \in \mathcal{E}_t$ representing an interaction between nodes $u, v \in \mathcal{V}_t$, at time $t \in \mathcal{T}$, our objective is to design a temporal GNN model that addresses the problem of learning effective representations for link prediction and node classification while incorporating a reject option to manage uncertainty in predictions.

1. Link Prediction: Given a temporal graph, i.e., source node, destination node, current timestamp, and historical interactions before t, predict the presence or absence of links at time t. Where the abstain mechanism aims to make a prediction or refuse to predict for a pair of nodes based on a coverage criterion.

2. Node Classification: Predict the label or state of a node in a temporal graph at a time t based on previous information, with the option to abstain from the prediction on certain nodes to ensure high confidence in the predictions made.

4 Method

In this section, we describe our approach for both link prediction and node classification in dynamic graphs in continuous time using the coverage-based method. A neural network architecture designed for dynamic graphs can be seen as a combination of an encoder and a decoder. The encoder serves to transform a dynamic graph into node embeddings, whereas the decoder utilizes these embeddings to make predictions. Our method considers temporal graph networks to incorporate a reject option, enhancing prediction and handling uncertainty effectively. The architecture is illustrated in Fig. 2.

4.1 Temporal Graph Encoder

Our approach is based on the concept of temporal graph networks, which are designed as an encoder-decoder pair for dynamic graph learning. The encoder maps a dynamic graph into the temporal node embeddings $\mathbf{z}_u(t)$, capturing the temporal interactions among the nodes. The encoder updates the node embeddings with each interaction, ensuring those reflect the graph's latest state. Few such models are memory-based TGN[32], MLP-based GraphMixer[7], graph transformer-based TCL[40], random-walk-based CAWN[41], SSM-based DyGMamba[9] etc.

4.2 Coverage-Based Dynamic Link Prediction

Our approach to link prediction uses the concept of abstention prediction[12] to manage the uncertainty inherent in predicting links in a dynamic graph employing an abstention model (f, q) where f is the prediction function and q is the abstention function. The abstention function $q : \mathcal{X} \to \{0, 1\}$, if it takes value one, then the prediction is kept on hold for a given pair of nodes based on a coverage criterion. We modify the decoder part of the model to incorporate a reject option

through an abstention prediction mechanism. This mechanism evaluates each potential link for its likelihood and the model's confidence in its prediction. By allowing the model to abstain from making predictions when the uncertainty exceeds a threshold θ, we ensure it only predicts links it is confident about. The abstention prediction objective is defined as follows:

$$\mathcal{L}^{\mathcal{E}_t}_{(f,q)} = \hat{r}(f, q|\mathcal{E}_t) + \lambda\Psi(c - \hat{\phi}(q|\mathcal{E}_t)) \tag{2}$$

where $\hat{r}(f, q|\mathcal{E}_t)$ is the empirical abstention risk, λ is a hyperparameter controlling the importance of the coverage constraint, $\Psi(b) = \max(0, b)^2$ is a quadratic penalty function that helps ensure that the model does not abstain excessively, and c is the target coverage rate. The empirical abstention risk ($\hat{r}(f, q|\mathcal{E}_t)$) and the empirical coverage ($\hat{\phi}(q|\mathcal{E}_t)$) are calculated as follows:

$$\hat{r}(f, q|\mathcal{E}_t) = \frac{\sum_{(u,v,t)\in\mathcal{E}_t} \ell(f(\mathbf{z}_u(t), \mathbf{z}_v(t)), y_{\mathcal{E}_t})(1 - q(\mathbf{z}_u(t), \mathbf{z}_v(t)))}{|\mathcal{E}_t|\ \hat{\phi}(q|\mathcal{E}_t)} \tag{3}$$

$$\hat{\phi}(q|\mathcal{E}_t) = \frac{1}{|\mathcal{E}_t|} \sum_{(u,v,t)\in\mathcal{E}_t} (1 - q(\mathbf{z}_u(t), \mathbf{z}_v(t))) \tag{4}$$

where ℓ is the *loss function* and $\mathbf{z}_u(t)$, $\mathbf{z}_v(t)$ are the temporal node embeddings for source and destination node for link prediction, respectively, and $y_{\mathcal{E}_t}$ is the binary label of link between two nodes if exists or not. $\hat{\phi}$, the empirical coverage rate of the abstention function q, calculates the rate at which the edges till time t are selected, where $|\mathcal{E}_t|$ is the number of edges. The overall training objective combines the abstention loss with an auxiliary loss. For our case, we use the standard binary cross-entropy as an auxiliary loss, denoted as $\mathcal{L}^{\mathcal{E}_t}_h$ computed on the same prediction task. This auxiliary loss is generated by a dedicated auxiliary prediction head, active only during training. Its purpose is to ensure the shared encoder learns from all training instances, including those the abstention head might abstain from. This approach prevents overfitting to only high-confidence samples and promotes more robust representations. The overall training objective is:

$$\mathcal{L}^{\mathcal{E}_t} = \alpha\mathcal{L}^{\mathcal{E}_t}_{(f,q)} + (1 - \alpha)\mathcal{L}^{\mathcal{E}_t}_h \tag{5}$$

Where α is a parameter controlling the trade-off between abstention and auxiliary losses.

The temporal graph encoder captures node interaction timing using a time encoder, which is incorporated into the temporal node embeddings. This time information is then passed to both the abstention and prediction functions, allowing the model to decide whether to abstain from making a prediction at a given time t.

4.3 Coverage-Based Dynamic Node Classification

For node classification or user state change prediction[22], we apply a framework similar to link prediction, using the abstention prediction mechanism to introduce a reject option. Node embedding $\mathbf{z}_u(t)$, derived from the temporal graph encoder, incorporates information from the node's interaction history and its neighbors to produce a comprehensive representation of its current state in the graph. It serves as input to a classification model that incorporates a coverage-based abstention mechanism. The classifier is designed to accurately predict node labels $y_t \in \mathcal{Y}$ while being able to abstain from predictions when uncertainty exceeds the threshold θ. The abstention prediction objective in this case is as follows:

$$\mathcal{L}^{\mathcal{V}_t}_{(f,q)} = \hat{r}(f, q|\mathcal{V}_t) + \lambda\Psi(c - \hat{\phi}(q|\mathcal{V}_t)) \tag{6}$$

Where the empirical abstention risk ($\hat{r}(f, q|\mathcal{V}_t)$) and the empirical coverage ($\hat{\phi}(q|\mathcal{V}_t)$) are calculated as follows:

$$\hat{r}(f, q|\mathcal{V}_t) = \frac{\sum_{u\in\mathcal{V}_t} \ell(f(\mathbf{z}_u(t)), y_t)(1 - q(\mathbf{z}_u(t))}{|\mathcal{V}_t|\ \hat{\phi}(q|\mathcal{V}_t)} \tag{7}$$

$$\hat{\phi}(q|\mathcal{V}_t) = \frac{1}{|\mathcal{V}_t|} \sum_{u\in\mathcal{V}_t} (1 - q(\mathbf{z}_u(t))) \tag{8}$$

where $|\mathcal{V}_t|$ is the number of nodes. The node classification model is trained by minimizing a loss function that balances the precision of the abstention prediction with the desired coverage level, which is given by:

$$\mathcal{L}^{\mathcal{V}_t} = \alpha\mathcal{L}^{\mathcal{V}_t}_{(f,q)} + (1 - \alpha)\mathcal{L}^{\mathcal{V}_t}_h \tag{9}$$

where $\mathcal{L}^{\mathcal{V}_t}_h$ is the auxiliary loss (binary cross-entropy).

4.4 Handling Extreme Class Imbalance for Dynamic Node Classification

Dealing with an extreme class imbalance in node classification within CTDGs presents significant challenges. Class imbalance adversely affects model performance, particularly for minority classes, which is crucial for many real-world applications. To address this, we introduce an approach that modifies the training process to better represent minority classes, thereby improving the model's performance. We modify the components of our training objective given in Equation (9).

Auxiliary Loss for Class Imbalance: We propose an auxiliary loss function that explicitly accounts for the disproportionate representation of classes by assigning a higher weight to the minority class, which is defined as:

$$\mathcal{L}^{\mathcal{V}_t}_h = \beta\mathcal{L}^{\mathcal{V}_t}_{h_{minor}} + \mathcal{L}^{\mathcal{V}_t}_{h_{major}} \tag{10}$$

where $\mathcal{L}^{\mathcal{V}_t}_{h_{minor}}$ and $\mathcal{L}^{\mathcal{V}_t}_{h_{major}}$ represent the loss for the minority and majority classes, respectively, and β is a weighing factor that amplifies the contribution of the minority class to the overall loss. In our experiments, we set hyperparameter $\beta > 1$, highlighting our focus on the minority class, given

its significantly lower representation (less than 0.2% in our dataset). This allows us to tune our model's performance, ensuring that it provides reliable predictions in the case of extreme class imbalance.

5 Experimental Setup

In this section, we discuss experimental setup, which comprises baselines, datasets used, evaluation tasks performed, metrics used, and implementation details.

5.1 Dataset Used

We use four distinct datasets for dynamic link prediction and two for dynamic node classification where labels are available. We have chosen popular datasets like Wikipedia[22], Reddit (Soical Media)[22], and also included challenging datasets like UN Trade (Economics) and Can. Parl. (Politics) for diversity. These datasets are publicly available [2] by DGB[29].

5.2 Baselines Used

Although our framework is compatible with various temporal graph models supported by DyGLib[43], we opted to utilize TGN[32] and GraphMixer[7] due to their widespread adoption as baselines.

TGN Encoder [32] : The key components of TGN Encoder are: a) **Memory module** maintains a compressed representation of each node's historical interactions. b) **Message function** computes the impact of new interactions on the state of the node. c) **Message aggregator** combines messages from multiple events involving the same node in a batch, improving the update process. d) **Memory updater** integrates new interaction messages into the node's memory. e) **Embedding module** generates the current embedding of a node using its memory and the memories of its neighbors.

GraphMixer Encoder [7]: GraphMixer presents a simple, straightforward, yet effective approach built upon two primary modules: a link encoder and a node encoder based on multilayer perceptrons (MLPs) without relying on complex GNN architectures. a) **Link-encoder** employs a fixed time-encoding functioncos$(\boldsymbol{\omega} t)$. It encodes temporal information of links, followed by an MLP-Mixer[36] to summarize this temporal link information effectively. b) **Node-encoder** utilizes neighbor mean-pooling to aggregate and summarize the features of nodes, capturing essential node identity and feature information for subsequent processing steps.

5.3 Evaluation Tasks and Metrics

We evaluate the performance of our proposed model in two primary tasks: link prediction and node classification in dynamic graph settings to enhance the practical applicability and trustworthiness of the predictions by following [29, 32, 41–43]. For the dynamic link prediction task, our objective is to predict the likelihood of a link forming between two nodes at a given time. This task is divided into two settings: transductive and inductive. The *transductive* setting focuses on predicting future links among nodes observed during training, while the *inductive* setting aims at predicting links between the nodes not seen during training. Following established protocols, we utilize a multilayer perceptron (MLP) to process concatenated node embeddings of source and destination and output link probabilities. Evaluation metrics for this task include average precision (AP) and area under the receiver operating characteristic curve (AUC-ROC), with random (rnd), historical (hist), and inductive (ind) negative sampling strategies (NSS)[29] employed to improve the evaluation since the latter two are more challenging. Random NSS randomly selects negative edges from all possible node pairs, risking easy examples and collisions with positive edges. Historical NSS selects negatives from previously observed but currently absent edges, testing prediction of edge re-occurrence. Inductive NSS focuses on edges seen during testing but not training, assessing model generalization to unseen edges.

The dynamic node classification task seeks to identify the label of a node at a specific time. Using the foundation set by the transductive dynamic link prediction model as a pre-trained model on a full dataset, we use a separate MLP for label prediction, adhering to the classification with a reject option mechanism to allow uncertainty handling. The datasets used for evaluation exclude those without dynamic node labels. The primary metric for this task is AUC-ROC, selected to address label imbalance issues commonly encountered in dynamic graphs[32, 42, 43].

For both tasks, we use a chronological data split strategy, allocating 70% of the data for training, 15% for validation, and 15% for testing to simulate real-world temporal dynamics and ensure a fair evaluation of our model's predictive capabilities.

5.4 Implementation Details

For the link prediction task, we optimize both the models by Adam optimizer[20] for all datasets. For the node classification task, we optimize both the models by Adam optimizer for the Wikipedia dataset and SGD for the Reddit dataset. We train the models for 75 epochs and use the early stopping strategy with a patience of 10. We select the model that achieves the best performance in the validation set for testing. We set the batch size to 200, λ to 32, α to 0.5, i.e., equal weight to both abstention prediction loss and auxiliary loss for all the methods on all the datasets. We perform the grid search to find the best settings of some critical hyperparameters. This step is conducted during training. We tune the learning rate using cross-validation. Even though the training process aims to meet the target coverage c via the loss function, the actual coverage achieved on the test set with a fixed threshold $e.g. \theta = 0.5$ may differ due to distribution shift. Therefore,

[2] https://zenodo.org/record/7213796#.Y1cO6y8r30o

Table 1. AP for transductive dynamic link prediction with random, historical, and inductive negative sampling strategies for various coverage rates. The best and second-best results are emphasized by **bold** and underlined.

NSS	Coverage (%)	TGN				GraphMixer			
		Wikipedia	Reddit	UN Trade	Can. Parl.	Wikipedia	Reddit	UN Trade	Can. Parl.
rnd	100	98.56 ± 0.06	98.63 ± 0.02	64.87 ± 1.93	74.14 ± 1.51	97.30 ± 0.05	97.35 ± 0.02	61.68 ± 0.16	79.96 ± 0.95
	90	99.33 ± 0.06	99.28 ± 0.07	71.67 ± 1.04	75.38 ± 1.44	98.24 ± 0.07	97.66 ± 0.06	68.57 ± 0.21	80.56 ± 1.71
	80	99.73 ± 0.08	99.54 ± 0.06	76.59 ± 1.50	77.21 ± 5.87	98.89 ± 0.06	98.41 ± 0.03	72.50 ± 1.24	82.44 ± 1.31
	70	99.85 ± 0.02	99.81 ± 0.04	79.09 ± 1.66	81.57 ± 7.38	99.61 ± 0.14	99.32 ± 0.08	71.64 ± 8.98	95.25 ± 0.65
	60	**99.88 ± 0.02**	99.86 ± 0.04	81.46 ± 1.81	**90.19 ± 3.50**	**99.68 ± 0.11**	99.52 ± 0.11	69.99 ± 11.27	**95.98 ± 0.57**
	50	99.87 ± 0.05	**99.91 ± 0.03**	**84.46 ± 0.95**	89.46 ± 6.67	99.63 ± 0.08	**99.65 ± 0.11**	**80.94 ± 2.11**	94.68 ± 1.25
hist	100	87.07 ± 0.81	80.63 ± 0.30	59.44 ± 2.22	70.92 ± 2.11	91.14 ± 0.16	77.50 ± 0.54	57.61 ± 1.11	80.66 ± 1.37
	90	90.81 ± 0.59	81.68 ± 1.07	70.36 ± 0.80	74.18 ± 1.94	94.25 ± 0.43	78.17 ± 0.25	63.36 ± 3.08	83.10 ± 1.57
	80	94.46 ± 0.64	83.78 ± 0.33	74.44 ± 3.55	72.82 ± 6.51	96.57 ± 0.63	80.57 ± 0.33	69.35 ± 3.68	84.08 ± 2.60
	70	95.99 ± 3.62	87.25 ± 1.76	71.94 ± 8.63	76.08 ± 9.00	97.99 ± 2.64	83.61 ± 1.96	75.01 ± 4.15	94.97 ± 0.55
	60	97.06 ± 3.84	88.66 ± 1.68	76.36 ± 2.16	86.97 ± 4.56	**98.56 ± 2.81**	85.32 ± 1.76	69.62 ± 1.57	**95.05 ± 0.97**
	50	**97.82 ± 4.50**	**89.27 ± 2.41**	**76.65 ± 2.85**	**87.34 ± 3.79**	97.40 ± 3.50	**85.91 ± 2.04**	**81.70 ± 2.52**	94.18 ± 1.33
ind	100	86.57 ± 0.96	88.02 ± 0.35	61.70 ± 2.21	68.05 ± 2.69	88.83 ± 0.13	85.21 ± 0.28	60.89 ± 1.01	77.74 ± 1.42
	90	89.75 ± 1.03	90.17 ± 0.82	72.31 ± 0.77	71.28 ± 2.90	92.11 ± 0.23	85.76 ± 0.34	67.57 ± 2.52	79.98 ± 2.47
	80	93.19 ± 0.57	92.89 ± 0.80	77.43 ± 2.94	69.06 ± 9.77	95.04 ± 0.37	88.74 ± 0.43	74.08 ± 1.72	79.26 ± 3.90
	70	95.85 ± 3.61	94.95 ± 2.27	75.75 ± 6.69	72.87 ± 10.05	96.45 ± 3.70	91.38 ± 0.86	79.47 ± 2.22	93.47 ± 0.56
	60	96.76 ± 4.00	**96.57 ± 2.68**	**78.65 ± 1.93**	84.91 ± 5.02	**97.66 ± 3.90**	93.11 ± 1.45	77.98 ± 1.54	94.11 ± 0.84
	50	**97.75 ± 4.63**	95.93 ± 3.33	78.30 ± 2.15	**85.12 ± 6.52**	94.79 ± 6.29	**94.35 ± 1.60**	**85.47 ± 2.57**	**94.28 ± 1.07**

to analyze the model's performance-reliability trade-off consistently, we evaluate at predefined coverage levels (e.g., 90%, 80%, etc.). For each target coverage level reported (e.g., 80%), we calculate the corresponding threshold θ_{80} by finding the 80th percentile of the uncertainty scores $a(x)$ produced by the model on the entire test set. Metrics are then computed solely on the test samples satisfying $a(x) \leq \theta_{80}$. The θ values range from 0 to 1. For node classification, we search for the best value of β in the range of [2,100] and get the best results when $\beta = 2$ or 5. Both encoders use 100D time encoding and 172D output representation. TGN uses 172D node memory with 2 graph attention heads, using GRU for memory updates. GraphMixer uses 2 MLP-Mixer layers with a time slot gap of 2000 interactions. We run each model five times with seeds from 0 to 4 and report the average performance and standard deviation to eliminate deviations. We use NVIDIA GeForce RTX 4090 with 24 GB memory for link prediction task. The GPU device used for the node classification task is NVIDIA GeForce RTX 2080 Ti with 11 GB memory.

6 Experimental Results

In this section, we report the performance of our proposed continuous-time dynamic graph neural network with a reject option and compare our results with baselines.

6.1 Results: Dynamic Link Prediction

We evaluated our rejection-aware framework on the dynamic link prediction task using both TGN and GraphMixer as base encoders, comparing performance against standard baselines run using DyGLib [42]. The primary goal was to assess the improvement in prediction reliability when allowing the model to abstain based on uncertainty.

Table 1 summarizes the transductive link prediction results, reporting Average Precision (AP) across all three negative sampling strategies (NSS). The 100% coverage rows represent the baseline performance without rejection. A consistent and significant trend is observed across all datasets, models, and NSS settings: as the coverage level decreases (meaning the model abstains on a larger fraction of uncertain predictions, from 90% down to 50% coverage), the AP score calculated on the remaining non-abstained predictions progressively improves.

This shows the effectiveness of the rejection mechanism in filtering out low-confidence predictions, thereby enhancing the precision of the accepted ones. For example, focusing on the challenging Canadian Parliament (Can. Parl.) dataset, which exhibits lower baseline scores, the improvement is notable. With our model, the AP/AUC score notably increases, emphasizing the model's efficiency in managing harder prediction environments. For example, using TGN with random NSS, the AP increases from a baseline of (1st row) at 100% coverage to 90.19% AP (5th row) at 60% coverage in Can. Parl. dataset, marking almost 16.05% improvement in AP. These gains show the value of abstention for boosting trustworthiness, particularly in environments with sparse or noisy interaction data typical of many real-world social or information networks.

Further illustrating the performance-reliability trade-off, Figure 3 plots the relationship between coverage and prediction performance (AUC) for the *inductive* link prediction task

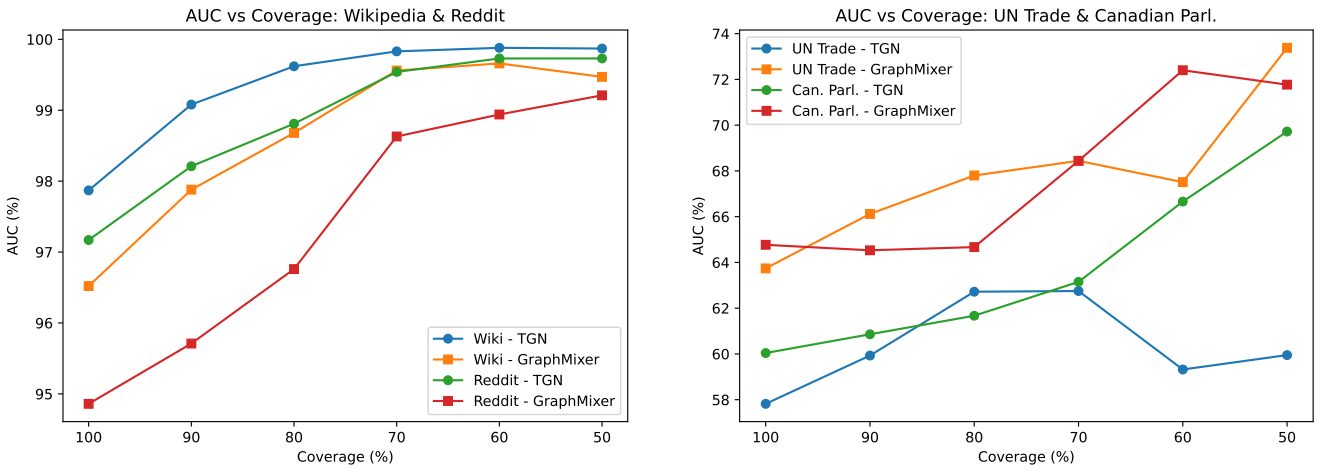

Figure 3. Performance (AUC %) vs. Coverage (%) Trade-off for Inductive Link Prediction in Random NSS.

(using Random NSS). Consistent with the AP results, the inductive AUC generally increases as coverage decreases (moving rightward on the x-axis). This confirms that the benefits of abstention—achieving higher accuracy on a more reliable subset of predictions—extend to the demanding inductive setting, where generalization to unseen graph elements is crucial. The visual trend highlights the model's capability to manage uncertainty effectively across different evaluation scenarios.

One observation is that AP/AUC increases to a certain point (peak) as coverage decreases. It starts decreasing as coverage goes on decreasing.

6.2 Results: Dynamic Node Classification

For dynamic node classification, we evaluate our model on the Wikipedia and Reddit datasets, as shown in Table 2. The initial results with 100% coverage use DyGLib, which establishes a benchmark. As we implement the reject option, progressively rejecting 10% of samples in each subsequent test until we reach 60% coverage, we observe a significant improvement in AUC scores (where $\beta = 1$). For example, in Table 2 we achieve 89.78% AUC (ninth row) in 60% coverage w.r.t. 86.23% AUC (first row) in 100% coverage for the TGN model on Wikipedia, marking almost 3.55% improvement in AUC.

This increase in AUC with decreasing coverage indicates that the model is effectively prioritizing more reliable predictions, as less certain predictions are rejected. Furthermore, due to the extreme imbalance in class distribution (minority class below 0.2%), we provide higher weightage to minority class in the auxiliary loss, which led to further improvements in AUC (where $\beta > 1$). For example, in Table 2, we get 69.58% AUC (tenth row) with $\beta = 5$ w.r.t.66.03% AUC (ninth row) in 60% coverage for TGN model on Reddit dataset. This marks almost 3.55% improvement in AUC while addressing the class imbalances. This strategy improves the efficacy of our approach in not only managing the reject option but also addressing extreme class imbalances, which are particularly challenging in dynamic environments.

Table 2. AUC for coverage-based dynamic node classification for various coverage rates with and without handling class imbalance. The best and second-best performing results are emphasized by **bold** and underlined fonts. * denotes β is 2, otherwise 5 for all other cases where $\beta > 1$.

Coverage(%)	β	TGN		GraphMixer	
		Wikipedia	Reddit	Wikipedia	Reddit
100	= 1	86.23 ± 3.30	63.08 ± 1.48	86.19 ± 1.10	64.81 ± 2.08
90	= 1	88.00 ± 2.00	64.81 ± 1.54	87.27 ± 1.72	65.26 ± 2.07
	> 1	88.44 ± 2.33*	65.12 ± 1.53	87.61 ± 1.06*	66.20 ± 1.88
80	= 1	88.78 ± 3.46	65.63 ± 1.31	87.93 ± 1.53	66.08 ± 2.99
	> 1	90.02 ± 2.29*	66.31 ± 3.26	88.05 ± 1.72	67.00 ± 2.40
70	= 1	89.48 ± 2.53	66.10 ± 2.34	87.79 ± 1.37	67.42 ± 3.27
	> 1	**90.64 ± 3.37**	67.90 ± 1.81	**88.38 ± 0.99***	**68.23 ± 2.96**
60	= 1	89.78 ± 2.76	66.03 ± 3.45	85.97 ± 2.62	67.23 ± 3.36
	> 1	89.86 ± 2.52	**69.58 ± 2.96**	84.83 ± 2.30*	67.87 ± 4.01

In real-world scenarios, we can't reject 40-50% predictions. But as we can see, with a decrease in coverage, performance keeps on increasing. Note that the model rejects examples; it does not predict very confidently. We have to find a balance between the rate of abstention and the model confidence based on the application domain, number of data points, etc. Overall, the experiments validate our hypothesis that integrating a reject option within CTDGs significantly enhances the reliability and accuracy of predictions in dynamic graphs. The clear improvement in performance metrics across different datasets, coverage settings, and tasks confirms the practical utility of our model in risk-sensitive applications, where precision and reliability are crucial.

7 Conclusion

The experimental findings convincingly demonstrate the efficacy of our proposed CTDG model with a reject option for both link prediction and node classification tasks. The model strategically abstains from making predictions when encountering high uncertainty, leading to a significant improvement in AUC/AP scores. This establishes the model's proficiency in managing the trade-off between prediction confidence and coverage. In the future, we can explore cost-based or other abstention methods to enhance the model's applicability in different contexts. Another future direction could involve demonstrating the model's performance in real-world applications, showcasing its practical relevance. Considering our method for another kind of graph model, such as discrete time temporal graphs, can also be explored. Overall, our work presents a compelling solution for dynamic graph applications that demand high precision and reliability, particularly in risk-sensitive domains. The model's ability to effectively manage uncertainty and class imbalance makes it a valuable tool for various real-world applications.

References

[1] Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. 2020. Adaptive Graph Convolutional Recurrent Network for Traffic Forecasting. In *NeurIPS*.

[2] Ali Behrouz and Farnoosh Hashemi. 2023. Learning Temporal Higher-order Patterns to Detect Anomalous Brain Activity. In *TGLW @ NeurIPS*.

[3] Ali Behrouz and Margo Seltzer. 2022. Anomaly Detection in Multiplex Dynamic Networks: from Blockchain Security to Brain Disease Prediction. In *TGLW @ NeurIPS*.

[4] Yuzhou Cao, Tianchi Cai, Lei Feng, Lihong Gu, Jinjie GU, Bo An, Gang Niu, and Masashi Sugiyama. 2022. Generalizing Consistent Multi-Class Classification with Rejection to be Compatible with Arbitrary Losses. In *NeurIPS*.

[5] Nontawat Charoenphakdee, Zhenghang Cui, Yivan Zhang, and Masashi Sugiyama. 2021. Classification with Rejection Based on Cost-sensitive Classification. In *ICML*. 1507–1517.

[6] Victor Chomel, Nathanaël Cuvelle-Magar, Maziyar Panahi, and David Chavalarias. 2022. Polarization identification on multiple timescale using representation learning on temporal graphs in Eulerian description. In *TGLW @ NeurIPS*.

[7] Weilin Cong, Si Zhang, Jian Kang, Baichuan Yuan, Hao Wu, Xin Zhou, Hanghang Tong, and Mehrdad Mahdavi. 2023. Do We Really Need Complicated Model Architectures For Temporal Networks?. In *ICLR*.

[8] Ed Davis, Ian Gallagher, Daniel John Lawson, and Patrick Rubin-Delanchy. 2025. Valid Conformal Prediction for Dynamic GNNs. In *ICLR*.

[9] Zifeng Ding, Yifeng Li, Yuan He, Antonio Norelli, Jingcheng Wu, Volker Tresp, Yunpu Ma, and Michael Bronstein. 2024. DyGMamba: Efficiently Modeling Long-Term Temporal Dependency on Continuous-Time Dynamic Graphs with State Space Models. *arXiv preprint arXiv:2408.04713* (2024).

[10] R. El-Yaniv and R. Wiener. 2007. On the foundations of cost-sensitive learning. *JMLR* (2007), 1393–1426.

[11] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *RLGM @ ICLR-W*.

[12] Yonatan Geifman and Ran El-Yaniv. 2019. SelectiveNet: A Deep Neural Network with an Integrated Reject Option. In *ICML*.

[13] William L Hamilton. 2020. *Graph representation learning*. Morgan & Claypool Publishers.

[14] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NIPS*. 1025–1035.

[15] Kexin Huang, Ying Jin, Emmanuel Candes, and Jure Leskovec. 2024. Uncertainty quantification over graph with conformalized graph neural networks. *NeurIPS* 36 (2024).

[16] Shenyang Huang, Yasmeen Hitti, Guillaume Rabusseau, and Reihaneh Rabbany. 2020. Laplacian Change Point Detection for Dynamic Graphs. In *ACM SIGKDD KDD*.

[17] Shenyang Huang, Farimah Poursafaei, Jacob Danovitch, Matthias Fey, Weihua Hu, Emanuele Rossi, Jure Leskovec, Michael M. Bronstein, Guillaume Rabusseau, and Reihaneh Rabbany. 2023. Temporal Graph Benchmark for Machine Learning on Temporal Graphs. In *NeurIPS Datasets and Benchmarks Track*.

[18] Bhavya Kalra, Kulin Shah, and Naresh Manwani. 2021. RISAN: Robust instance specific deep abstention network. In *UAI*.

[19] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. 2020. Representation Learning for Dynamic Graphs: A Survey. *JMLR* (2020), 1–73.

[20] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

[21] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[22] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks. In *KDD*. 1269–1278.

[23] Naresh Manwani, Kalpit Desai, Sanand Sasidharan, and Ramasubramanian Sundararajan. 2015. Double ramp loss based reject option classifier. In *PAKDD*. 151–163.

[24] Pritam Nath, Govind Waghmare, Nancy Agrawal, Nitish Kumar, and Siddhartha Asthana. 2023. TBoost: Gradient Boosting Temporal Graph Neural Networks. In *TGLW @ NeurIPS*.

[25] Chenri Ni, Nontawat Charoenphakdee, Junya Honda, and Masashi Sugiyama. 2019. On the Calibration of Multiclass Classification with Rejection. In *NeurIPS*, Vol. 32.

[26] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. 2020. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *AAAI*. 5363–5370.

[27] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS-W*.

[28] James W Pennebaker, Martha E Francis, and Roger J Booth. 2001. Linguistic inquiry and word count: LIWC 2001. *Mahway: Lawrence Erlbaum Associates* (2001), 2001.

[29] Farimah Poursafaei, Andy Huang, Kellin Pelrine, and Reihaneh Rabbany. 2022. Towards Better Evaluation for Dynamic Link Prediction. In *NeurIPS Datasets and Benchmarks Track*.

[30] Shivshankar Reddy, Pranav Poduval, Anand Vir Singh Chauhan, Maneet Singh, Sangam Verma, Karamjit Singh, and Tanmoy Bhowmik. 2021. Tegraf: temporal and graph based fraudulent transaction detection framework. In *ACM International Conference on AI in Finance*. 1–8.

[31] Jakub Reha, Giulio Lovisotto, Michele Russo, Alessio Gravina, and Claas Grohnfeldt. 2023. Anomaly Detection in Continuous-Time Temporal Provenance Graphs. In *TGLW @ NeurIPS*.

[32] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. 2020. Temporal Graph Networks for Deep Learning on Dynamic Graphs. In *GRL @ ICML-W*.

[33] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The Graph Neural Network Model. *IEEE Transactions on Neural Networks* (2009), 61–80.

[34] Kulin Shah and Naresh Manwani. 2019. Sparse reject option classifier using successive linear programming. In *AAAI*. 4870–4877.
[35] Yuxing Tian, Yiyan Qi, and Fan Guo. 2024. FreeDyG: Frequency Enhanced Continuous-Time Dynamic Graph Model for Link Prediction. In *ICLR*.
[36] Ilya O. Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. 2021. MLP-Mixer: An all-MLP Architecture for Vision. In *NeurIPS*. 24261–24272.
[37] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. 2019. DyRep: Learning Representations over Dynamic Graphs. In *ICLR*.
[38] Sai Supriya Varugunda, Ching-Hao Fan, and Lijing Wang. 2023. Exploring Graph Structure in Graph Neural Networks for Epidemic Forecasting. In *TGLW @ NeurIPS*.
[39] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.
[40] Lu Wang, Xiaofu Chang, Shuang Li, Yunfei Chu, Hui Li, Wei Zhang, Xiaofeng He, Le Song, Jingren Zhou, and Hongxia Yang. 2021. TCL: Transformer-based Dynamic Graph Modelling via Contrastive Learning. *CoRR* (2021).
[41] Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. 2021. Inductive Representation Learning in Temporal Networks via Causal Anonymous Walks. In *ICLR*.
[42] Da Xu, Chuanwei Ruan, Evren Körpeoglu, Sushant Kumar, and Kannan Achan. 2020. Inductive representation learning on temporal graphs. In *ICLR*.
[43] Le Yu, Leilei Sun, Bowen Du, and Weifeng Lv. 2023. Towards Better Dynamic Graph Learning: New Architecture and Unified Library. In *NeurIPS*.

A Datasets

We use four publicly available datasets in our experiments, detailed below. The dataset statistics are shown in Table 3.

- **Reddit [22]:** A month's worth of posts from users across the 984 most active subreddits, resulting in a total of 672,447 interactions from the 10,000 most active users. Each post's content is transformed into a feature vector using LIWC [28]. This dataset also includes dynamic labels indicating whether users are banned from posting, with 366 true banned labels, marking a proportion of 0.05% of the interactions.
- **Wikipedia [22]:** Comprising one month of Wikipedia edits, this dataset captures interactions from users who have made edits on the 1,000 most edited pages, totaling 157,474 interactions from 8,227 users. Each interaction is represented as a 172-dimensional LIWC feature vector. Additionally, dynamic labels are provided to indicate temporary bans of users, with 217 positive banned labels among the interactions, equating to 0.14%.
- **Canadian Parliament (Can. Parl.) [16]:** This dynamic political network dataset records interactions between Members of Parliament (MPs) in Canada from 2006 to 2019, where nodes represent MPs and links indicate mutual "yes" votes on bills. The weight of a link reflects the frequency of mutual affirmative votes within a year.
- **UN Trade [29]:** Spanning over 30 years, this dataset documents food and agriculture trade between 181 nations. The links between nations are weighted by the normalized value of agriculture imports or exports, showing the intensity of trade relationships.

Table 3. Datasets. #N&L Feat. denotes node and link feature dimension.

Datasets	#Nodes	#Links	#N&L Feat.	Bipartite	Time Granularity	Label
Wikipedia	9,227	157,474	– & 172	True	Unix	2
Reddit	10,984	672,447	– & 172	True	Unix	2
Can. Parl.	734	74,478	– & 1	False	Years	–
UN Trade	255	507,497	– & 1	False	Years	–

B Hyperparameters

All three encoders use 100-dimensional time encoding and 172-dimensional output representation. TGN uses a 172-dimensional node memory and uses 2 graph attention heads like DyGFormer, with memory updates managed by Gated Recurrent Units (GRU). GraphMixer does not use node memory but instead has 2 MLP-Mixer layers and operates with a time gap of 2000. DyGFormer introduces a 50-dimensional neighbor co-occurrence encoding alongside the time encoding and employs 2 Transformer layers to process the input.

Table 5. Compute Statistics for Can. Parl. dataset on link prediction

TG Encoder	Train Time (Sec/epoch)	Test Time (Sec)	GPU Usage (MB)
TGN	9	1	668
TGAT	75	22	2800
GraphMixer	27	6	1708
CAWN	161	36	12114
DyGFormer	91	20	18326
TCL	15	1	1616
Jodie	4	1	640

We perform the grid search to find the best settings of some critical hyperparameters. However, this step is conducted during training. Once the model is trained, the number of covered examples on the test dataset will vary significantly as θ changes because the model has not seen the test data during training. Nevertheless, with the selection scores for each interaction available, we sorted them and selected a θ value that aligns with the desired coverage. The θ values range from 0 to 1. As the coverage decreases to 90%, 80%, and so on down to 50%, the uncertainty threshold θ also decreases accordingly.

We experimented with a wide range of β values as a hyperparameter and found that when β the value is less than 10, it produces a better result. We're attaching the following table for TGN on the Wikipedia dataset to show the variation AUC w.r.t. different values of beta. Below is the comparison between β vs. AUC shown in Table 4.

Table 4. Effect of β on AUC at coverage 90%, 80% and 70%, respectively for TGN encoder on Wikipedia dataset

β	AUC@90	AUC@80	AUC@70
1	88	88.78	89.48
2	88.44	90.02	90.28
5	87.96	89.7	90.64
10	87.56	89.14	89.98
25	87.67	88.29	89.42
100	86.84	88.32	88.74

C Compute Statistics

We have compared compute statistics for different temporal graph encoders for the Can. Parl. dataset in table 5 on the link prediction task. We train a maximum of 100 epochs for training. Based on the following table, we have chosen to use TGN and GraphMixer as our baseline.

D Additional Results

Tables 6 and 7 present the link prediction results for inductive AP and transductive AUC-ROC, respectively. We have added link prediction results for DyGFormer[43] on all four datasets in Table 8 for the transductive setting and Table 9 for the inductive setting. We have also added node classification results for DyGFormer in Table 10.

Table 6. AP for inductive dynamic link prediction with random, historical, and inductive negative sampling strategies for various coverage rates. The best and second-best performing results are emphasized by **bold** and underlined fonts.

NSS	Coverage (%)	TGN				GraphMixer			
		Wikipedia	Reddit	UN Trade	Can. Parl.	Wikipedia	Reddit	UN Trade	Can. Parl.
rnd	100	97.98 ± 0.08	97.34 ± 0.05	56.55 ± 2.24	57.21 ± 1.22	96.82 ± 0.06	95.13 ± 0.01	61.83 ± 0.09	59.38 ± 1.05
	90	99.08 ± 0.06	98.23 ± 0.14	63.23 ± 1.72	57.08 ± 2.88	98.03 ± 0.12	95.76 ± 0.14	68.16 ± 0.19	58.96 ± 0.85
	80	99.57 ± 0.07	98.79 ± 0.10	**66.04 ± 3.86**	57.77 ± 4.64	98.76 ± 0.08	96.76 ± 0.12	70.97 ± 1.19	58.83 ± 0.99
	70	99.77 ± 0.05	99.45 ± 0.09	65.51 ± 2.31	64.57 ± 8.02	99.49 ± 0.14	98.39 ± 0.23	70.54 ± 7.44	70.65 ± 3.30
	60	**99.82 ± 0.06**	99.68 ± 0.04	61.23 ± 5.83	69.82 ± 6.54	**99.59 ± 0.12**	98.75 ± 0.25	68.45 ± 9.95	76.29 ± 4.96
	50	99.80 ± 0.08	**99.73 ± 0.04**	60.69 ± 5.56	**71.55 ± 7.14**	99.46 ± 0.09	**99.03 ± 0.29**	**78.39 ± 0.90**	**76.44 ± 2.39**
hist	100	82.12 ± 1.37	61.59 ± 0.48	52.96 ± 1.27	56.75 ± 1.29	87.63 ± 0.28	61.73 ± 0.36	56.41 ± 0.95	58.64 ± 0.77
	90	84.97 ± 1.13	61.84 ± 0.66	59.12 ± 3.88	55.95 ± 2.40	91.30 ± 0.47	62.42 ± 0.29	60.48 ± 2.44	58.70 ± 0.90
	80	88.85 ± 0.83	63.02 ± 0.89	**61.20 ± 4.00**	56.47 ± 4.78	94.22 ± 1.03	63.66 ± 0.27	65.74 ± 4.66	58.85 ± 1.53
	70	92.42 ± 4.94	66.99 ± 4.34	58.80 ± 5.99	65.27 ± 9.09	96.01 ± 3.95	66.98 ± 3.44	74.08 ± 3.63	73.00 ± 2.67
	60	94.81 ± 5.67	67.96 ± 3.89	58.90 ± 4.34	70.98 ± 6.00	**97.33 ± 4.50**	68.73 ± 3.33	70.64 ± 1.50	77.83 ± 4.37
	50	**96.63 ± 6.68**	**68.83 ± 6.00**	59.02 ± 5.16	**71.27 ± 8.94**	94.79 ± 6.78	**69.43 ± 3.38**	**82.22 ± 1.63**	**78.41 ± 1.74**
ind	100	82.12 ± 1.38	61.59 ± 0.48	52.90 ± 1.25	56.78 ± 1.31	87.63 ± 0.28	61.74 ± 0.35	56.35 ± 0.93	58.63 ± 0.79
	90	84.98 ± 1.13	61.85 ± 0.66	59.06 ± 3.89	55.97 ± 2.41	91.30 ± 0.47	62.44 ± 0.28	60.31 ± 2.54	58.72 ± 0.92
	80	88.86 ± 0.82	63.01 ± 0.90	**61.16 ± 3.98**	56.53 ± 4.87	94.22 ± 1.03	63.69 ± 0.26	65.64 ± 4.79	58.89 ± 1.61
	70	92.45 ± 4.95	66.98 ± 4.35	58.78 ± 5.95	65.39 ± 9.15	96.01 ± 3.95	67.00 ± 3.44	74.13 ± 3.50	73.01 ± 2.65
	60	94.80 ± 5.68	67.98 ± 3.88	58.83 ± 4.36	70.90 ± 6.02	**97.33 ± 4.51**	68.76 ± 3.31	70.65 ± 1.47	77.81 ± 4.34
	50	**96.64 ± 6.69**	**68.84 ± 5.99**	58.98 ± 5.22	**71.21 ± 8.90**	94.79 ± 6.78	**69.48 ± 3.35**	**82.16 ± 1.72**	**78.45 ± 1.73**

Table 7. AUC for transductive dynamic link prediction with random, historical, and inductive negative sampling strategies for various coverage rates. The best and second-best performing results are emphasized by **bold** and underlined fonts.

NSS	Coverage (%)	TGN				GraphMixer			
		Wikipedia	Reddit	UN Trade	Can. Parl.	Wikipedia	Reddit	UN Trade	Can. Parl.
rnd	100	98.48 ± 0.07	98.59 ± 0.02	67.74 ± 1.66	77.94 ± 1.55	97.01 ± 0.07	97.20 ± 0.02	64.89 ± 0.21	84.64 ± 0.57
	90	99.32 ± 0.05	99.31 ± 0.06	70.24 ± 0.92	80.17 ± 1.44	98.04 ± 0.08	97.63 ± 0.07	68.01 ± 0.21	85.39 ± 1.04
	80	99.75 ± 0.09	99.57 ± 0.06	74.50 ± 1.21	82.32 ± 2.21	98.72 ± 0.07	98.40 ± 0.03	70.52 ± 0.70	86.77 ± 0.94
	70	99.88 ± 0.06	99.85 ± 0.02	76.71 ± 1.64	82.05 ± 5.10	99.63 ± 0.28	99.41 ± 0.06	71.36 ± 4.07	93.05 ± 1.10
	60	99.90 ± 0.04	99.90 ± 0.02	79.25 ± 1.86	88.03 ± 3.69	**99.72 ± 0.22**	99.59 ± 0.10	69.69 ± 7.48	**94.13 ± 0.94**
	50	**99.90 ± 0.02**	**99.92 ± 0.01**	**82.14 ± 1.18**	**89.69 ± 2.49**	99.61 ± 0.19	**99.72 ± 0.05**	**77.15 ± 2.89**	92.91 ± 1.43
hist	100	83.01 ± 0.60	80.92 ± 0.18	63.78 ± 2.37	73.64 ± 2.14	88.00 ± 0.22	77.31 ± 0.27	63.40 ± 1.06	83.56 ± 2.13
	90	89.27 ± 0.41	84.50 ± 0.80	68.93 ± 0.88	77.51 ± 0.72	92.46 ± 0.57	79.94 ± 0.83	66.35 ± 1.82	86.34 ± 1.15
	80	94.43 ± 0.67	87.24 ± 0.50	72.87 ± 2.63	77.12 ± 2.51	95.50 ± 0.67	83.49 ± 0.59	69.23 ± 1.06	85.54 ± 2.00
	70	95.23 ± 6.09	89.86 ± 2.21	72.42 ± 4.33	76.69 ± 5.84	97.35 ± 4.18	85.97 ± 2.10	72.67 ± 1.16	**92.43 ± 1.03**
	60	96.25 ± 6.11	**91.55 ± 2.31**	74.73 ± 1.93	83.84 ± 4.56	**97.87 ± 4.38**	87.94 ± 2.70	70.61 ± 1.15	92.35 ± 1.68
	50	**96.81 ± 6.82**	90.91 ± 2.69	**75.39 ± 1.98**	**86.68 ± 0.89**	95.94 ± 5.50	**89.17 ± 2.65**	**76.47 ± 3.75**	92.23 ± 1.42
ind	100	82.76 ± 0.71	84.58 ± 0.33	66.06 ± 2.26	70.93 ± 2.25	84.93 ± 0.27	82.31 ± 0.24	67.06 ± 0.81	80.88 ± 1.28
	90	88.18 ± 0.59	89.34 ± 0.55	70.75 ± 0.93	75.64 ± 2.36	89.76 ± 0.39	84.96 ± 0.91	69.81 ± 1.52	83.37 ± 1.56
	80	93.18 ± 0.53	92.79 ± 0.93	75.36 ± 2.52	73.70 ± 5.28	93.81 ± 0.37	88.71 ± 0.33	73.27 ± 1.02	82.50 ± 2.33
	70	95.14 ± 5.94	94.59 ± 4.28	75.28 ± 3.59	74.17 ± 5.77	95.52 ± 5.89	91.16 ± 2.90	77.07 ± 0.54	90.82 ± 0.85
	60	96.07 ± 6.15	**96.36 ± 4.84**	**77.08 ± 1.77**	82.36 ± 4.60	**96.78 ± 6.04**	93.10 ± 3.57	77.74 ± 1.66	91.40 ± 1.37
	50	**96.80 ± 6.84**	94.63 ± 5.72	76.93 ± 1.37	**85.01 ± 2.03**	92.49 ± 9.71	**94.31 ± 3.73**	**81.14 ± 4.06**	**92.10 ± 1.31**

Table 8. AP and AUC score for transductive dynamic link prediction with **DyGFormer** with random, historical, and inductive negative sampling strategies for various coverage rates. The best and second-best performing results are emphasized by **bold** and underlined fonts.

NSS	Coverage (%)	AP				AUC			
		Wikipedia	Reddit	UN Trade	Can. Parl.	Wikipedia	Reddit	UN Trade	Can. Parl.
rnd	100	99.11 ± 00.02	99.21 ± 00.01	65.37 ± 00.29	98.32 ± 00.19	99.01 ± 00.01	99.14 ± 00.01	69.06 ± 00.37	98.01 ± 00.18
	90	99.60 ± 00.03	99.64 ± 00.05	61.02 ± 05.45	99.04 ± 00.48	99.57 ± 00.04	99.62 ± 00.06	64.72 ± 03.46	98.81 ± 00.62
	80	99.86 ± 00.04	99.78 ± 00.05	**67.45 ± 07.22**	99.32 ± 00.22	99.86 ± 00.07	99.81 ± 00.09	69.30 ± 03.73	99.10 ± 00.41
	70	99.86 ± 00.05	99.87 ± 00.02	**77.76 ± 01.59**	99.05 ± 00.44	99.92 ± 00.05	99.93 ± 00.01	**75.45± 0.03**	98.49 ± 00.81
	60	**99.93 ± 00.01**	**99.88 ± 00.01**	62.40 ± 11.67	99.52 ± 00.52	**99.96 ± 00.00**	**99.94 ± 00.01**	65.46 ± 09.23	99.31 ± 00.83
	50	99.92 ± 00.02	99.13 ± 01.11	60.21 ± 03.03	**99.72 ± 00.23**	99.96 ± 00.01	99.65 ± 00.42	66.19 ± 04.06	**99.67 ± 00.27**
hist	100	74.60 ± 05.62	81.39 ± 01.49	60.73 ± 02.84	98.57 ± 00.12	75.00 ± 02.99	80.56 ± 00.46	70.70 ± 03.06	98.28 ± 00.18
	90	74.39 ± 03.53	84.42 ± 02.18	58.81 ± 02.81	98.97 ± 00.70	78.67 ± 02.87	86.48 ± 01.20	63.94 ± 03.04	98.68 ± 00.93
	80	**82.89 ± 08.86**	86.12 ± 02.04	70.69 ± 08.71	99.33 ± 00.28	85.53 ± 06.40	90.18 ± 01.19	**71.99 ± 05.76**	99.09 ± 00.48
	70	80.86 ± 08.52	87.13 ± 00.92	**80.53± 7.57**	98.97 ± 00.47	**85.89 ± 07.03**	92.21 ± 00.37	75.74 ± 9.98	98.31 ± 00.91
	60	71.71 ± 16.94	88.35 ± 02.93	57.65 ± 10.77	99.47 ± 00.55	79.09 ± 15.62	**92.99 ± 01.11**	58.43 ± 09.18	99.22 ± 00.90
	50	77.36 ± 21.58	**89.47 ± 01.96**	58.31 ± 04.63	**99.67 ± 00.28**	82.18 ± 23.39	89.45 ± 05.38	62.83 ± 06.28	**99.60 ± 00.33**
ind	100	67.87 ± 12.41	91.17 ± 00.79	56.47 ± 01.79	98.66 ± 00.10	69.88 ± 07.37	86.52 ± 00.82	65.25 ± 02.23	98.37 ± 00.16
	90	64.93 ± 06.60	95.11 ± 00.92	60.74 ± 03.45	99.20 ± 00.46	71.65 ± 05.05	93.89 ± 00.85	66.49 ± 03.17	99.00 ± 00.60
	80	76.77 ± 09.48	97.72 ± 00.95	72.23 ± 08.04	99.45 ± 00.29	**80.53 ± 07.03**	97.92 ± 01.19	74.00 ± 05.45	99.26 ± 00.48
	70	72.54 ± 15.14	98.72 ± 00.81	**82.98± 6.71**	99.05 ± 00.45	78.35 ± 13.70	99.25 ± 00.39	79.65± 06.71	98.50 ± 00.83
	60	68.53 ± 21.58	**99.38 ± 00.38**	64.11 ± 11.79	99.53 ± 00.53	73.34 ± 20.90	**99.64 ± 00.14**	67.77 ± 09.98	99.33 ± 00.85
	50	**77.02 ± 28.63**	95.00 ± 05.90	69.00 ± 05.72	**99.74 ± 00.21**	77.23 ± 31.64	92.84 ± 09.24	**75.88 ± 05.65**	**99.69 ± 00.26**

Table 9. AP and AUC score for inductive dynamic link prediction by **DyGFormer** with random, historical, and inductive negative sampling strategies for various coverage rates. The best and second-best performing results are emphasized by **bold** and underlined fonts.

NSS	Coverage (%)	AP				AUC			
		Wikipedia	Reddit	UN Trade	Can. Parl.	Wikipedia	Reddit	UN Trade	Can. Parl.
rnd	100	98.66 ± 00.04	98.73 ± 00.03	63.03 ± 00.26	93.13 ± 00.57	98.57 ± 00.02	98.58 ± 00.03	65.90 ± 00.32	91.36 ± 00.86
	90	99.38 ± 00.06	99.32 ± 00.08	59.56 ± 04.65	94.73 ± 00.87	99.35 ± 00.04	99.24 ± 00.09	62.68 ± 02.80	92.59 ± 00.69
	80	99.77 ± 00.04	99.66 ± 00.11	66.13 ± 08.17	95.93 ± 00.88	99.81 ± 00.07	99.66 ± 00.16	67.06± 04.76	94.09 ± 01.24
	70	99.85 ± 00.05	99.81 ± 00.01	**76.05 ± 1.44**	96.15 ± 00.22	99.89 ± 00.07	99.89 ± 00.01	**73.26 ± 2.64**	94.26 ± 00.29
	60	99.82 ± 00.04	**99.81 ± 00.01**	60.29 ± 11.96	96.76 ± 01.56	99.90 ± 00.02	**99.89 ± 00.01**	62.11 ± 09.78	95.46 ± 02.24
	50	**99.85 ± 00.02**	98.76 ± 01.44	56.92 ± 02.35	**97.24 ± 01.56**	**99.92 ± 00.01**	99.44 ± 00.66	61.56 ± 03.49	**95.97 ± 02.34**
hist	100	63.13 ± 09.13	61.80 ± 01.32	53.48 ± 01.52	93.41 ± 00.54	64.28 ± 06.05	62.78 ± 00.26	60.61 ± 02.21	91.79 ± 00.72
	90	61.22 ± 05.53	63.41 ± 01.09	58.02 ± 04.60	95.05 ± 00.88	65.74 ± 04.76	65.75 ± 00.30	62.86 ± 03.68	93.09 ± 00.65
	80	**69.87 ± 06.59**	63.68 ± 00.98	65.44 ± 06.97	96.29 ± 00.68	**73.77 ± 05.50**	67.94 ± 00.53	66.95 ± 04.56	94.54 ± 01.04
	70	67.35 ± 12.76	64.18 ± 01.47	**79.83 ± 6.85**	96.31 ± 00.27	71.72 ± 12.03	**70.39 ± 00.87**	**75.73 ± 9.28**	94.43 ± 00.39
	60	61.81 ± 14.01	65.11 ± 01.81	59.21 ± 10.25	96.89 ± 01.42	67.78 ± 16.27	70.18 ± 02.23	61.19 ± 08.50	95.60 ± 02.11
	50	66.38 ± 19.47	**69.21 ± 04.52**	61.33 ± 03.75	**97.35 ± 01.50**	70.91 ± 26.14	69.35 ± 06.52	67.26 ± 04.67	**96.14 ± 02.24**
ind	100	63.13 ± 09.15	61.77 ± 01.32	53.51 ± 01.51	93.42 ± 00.54	64.28 ± 06.05	62.76 ± 00.26	60.65 ± 02.20	91.81 ± 00.72
	90	61.19 ± 05.52	63.38 ± 01.09	58.04 ± 04.59	95.06 ± 00.88	65.72 ± 04.77	65.73 ± 00.30	62.88 ± 03.66	93.10 ± 00.65
	80	**69.86 ± 06.60**	63.66 ± 00.97	68.95 ± 06.27	96.30 ± 00.68	**73.76 ± 05.49**	67.92 ± 00.53	69.82 ± 04.54	94.55 ± 01.04
	70	67.34 ± 12.78	64.14 ± 01.46	**79.85 ± 6.86**	96.32 ± 00.27	71.72 ± 12.04	**70.36 ± 00.86**	**75.76 ± 9.28**	94.43 ± 00.39
	60	61.80 ± 14.02	65.09 ± 01.80	59.23 ± 10.25	96.90 ± 01.41	67.78 ± 16.27	70.15 ± 02.24	61.23 ± 08.51	95.61 ± 02.11
	50	66.40 ± 19.52	**69.19 ± 04.50**	61.34 ± 03.75	**97.35 ± 01.51**	70.92 ± 26.20	69.33 ± 06.48	67.27 ± 04.66	**96.13 ± 02.25**

Table 10. AUC for coverage-based dynamic node classification by **DyGFormer** for various coverage rates, with and without handling class imbalance. The best and second-best performing results are emphasized in **bold** and underlined fonts, respectively.

Coverage (%)	β	DyGFormer
		Wikipedia
100	$= 1$	87.04 ± 1.08
90	$= 1$	87.19 ± 0.86
	$= 2$	88.02 ± 1.87
80	$= 1$	88.47 ± 3.32
	$= 2$	<u>88.97 ± 2.37</u>
70	$= 1$	86.96 ± 3.60
	$= 2$	**89.17 ± 4.72**
60	$= 1$	83.79 ± 3.76
	$= 2$	83.42 ± 2.71