Fusion of GNN and GBDT Models for Graph and Node Classification

Muhammad Farhan, Nadeem Iqbal Kajla, Malik Daler Ali Awan, Mickaël Coustaty, Muhammad Muzzamil Luqman, Souhail Bakkali

Fusion of GNN and GBDT Models for Graph and Node Classification

Muhammad Farhan[1], Nadeem Iqbal Kajla[2,3(✉)], Malik Daler Ali Awan[1], Mickaël Coustaty[4], Muhammad Muzzamil Luqman[4], and Souhail Bakkali[4]

[1] Department of Software Engineering, Faculty of Computing, The Islamia University of Bahawalpur, Bahawalpur, Pakistan
[2] School of Computer Science, University of Galway, Galway, Ireland
[3] Institute of Computing, MNS University of Agriculture, Multan, Pakistan
nadeem.iqbal@mnsuam.edu.pk
[4] L3i, La Rochelle University, La Rochelle, France
mickael.coustaty@univ-lr.fr

Abstract. The discipline of graph-based machine learning, which focuses on learning from structured graph data, is expanding rapidly. Numerous applications in recommendation systems, bioinformatics, and social network analysis fall within this domain. However, traditional Graph Neural Networks (GNNs) face difficulties when dealing with datasets that frequently contain structured and graph data. Our approach addresses this challenge by creating a proposed fusion model of GNN and Gradient Boosting Decision Trees (GBDTs). We investigated the effectiveness of using the GNN and GBDT based fusion model using logistic regression by combining the embeddings of GNN and GBDT, stacking the predictions from GBDT variants for node classification and graph classification. The generality of the model is tested and validated on one heterogeneous and two homogeneous state-of-the-art datasets with average accuracy of A: Freebase: 0.6875, B: Letters (Low: 96.62, Med: 84.81, High: 78.06), C:Fingerprints:80.45, D:OGBG-MolHIV:89.10 outperforming individual methods. The results indicated that the fusion approach was effective in accurately classifying complete graphs and nodes, although their performance varied depending on the dataset and the characteristics of the graph being analyzed. This shows that the applied technique can get the appropriate results. The supplementary material of our work is publicly available at (https://github.com/mr49online/fusion_model).

Keywords: Fusion model · GBDT · GNN · Homogeneous graphs · Heterogeneous graphs · Graph classification · Node classification

1 Introduction

A homogeneous graph considers all nodes of similar type, and all edges denote the same type of relationship. For example, a social network can be characterized

L. Brun et al. (Eds.): GbRPR 2025, LNCS 15727, pp. 167–178, 2025.
https://doi.org/10.1007/978-3-031-94139-9_16

as a homogeneous graph, with nodes meaning people, and edges showing their connections. In contrast, a heterogeneous graph includes nodes and edges of numerous types, allowing the depiction of complex relationships between various entities. An example of a heterogeneous graph is a marketplace graph, which includes nodes such as buyers, sellers, and products, connected by edges like "wants-to-buy" or "has-bought." These distinctions are essential for modeling real-world systems with varying levels of complexity [1,2].

Graph Neural Networks (GNNs) [3,4] have emerged as a fundamental technology in graph mining, representing significant success in applications such as molecular design, computer vision, and recommendation systems. The important tasks in this domain are graph classification [5], which involves predicting the class of a graph based on its structural features, and node classification, where the goal is to predict the labels for nodes based on their features and relationships, with applications in bioinformatics, social network analysis, and chemistry. Recent approaches to this problem include GNNs, which excel at learning graph structures, and gradient boosting decision trees (GBDTs), known for capturing complex non-linear relationships. Despite their strengths, both methods face limitations in certain graph types.

Unlike homogeneous graphs, heterogeneous graphs require complex structures and advanced preprocessing to fully exploit their potential. Gradient-boosted decision trees (GBDT) [6] are highly effective for high-dimensional tabular data, and prior studies like BGNN [7], EBBS [8] and TreeXGNN [9] attempt to bridge GBDTs and GNNs. The TreeXGNN technique was only implemented for node classification task However, to the best of our knowledge, their application to heterogeneous graphs remains unexplored. A new attempt to combine GBDT with Heterogeneous Graph Neural Network HGNN was made in TabGraphs [10]. They contributed benchmark dataset, called TabGraphs, which includes several real world datasets. But this dataset also supports node classification and node regression tasks.

Motivation. As GNN performs well on only homogeneous graph datasets. GBDT performs well on heterogeneous tabular datasets. There is a dire need to make such models which have the best properties of both models and can be applicable to both homogeneous and heterogeneous datasets.

Contribution

- We purpose a modular and extensible fusion framework that combines graph embeddings from multiple GNNs (GCN and MPNN). These embeddings are combined with GBDT classifiers using a late fusion strategy.
- We perform comprehensive evaluations, including ablation studies and benchmarking on the diverse dataset, including OGBG-MolHIV, demonstrating the effectiveness of our approach.

2 Related Work

Deep learning models called Graph Convolutional Networks (GCN) and Message Passing Neural Networks (MPNN) are made especially for graph-structured data. Introduced by Kipf and Welling [11], GCNs aggregate information from neighbors of a node using a spectral technique, extending convolutional processes to graphs. Gilmer et al. [12] proposed MPNNs, which iteratively update node representations and pass messages between nodes to provide a generalized framework for graph-based learning. These models are appropriate for tasks like node classification and link prediction because they are excellent at capturing relational and structural information. On the other hand, ensemble techniques called gradient-boosted decision trees (GBDT), such as XGBoost [13], construct decision trees sequentially in order to reduce prediction errors. Despite their strength in tabular data, GBDTs are not naturally able to properly model graph-structured data since they do not take relational connections between entities into consideration.

GCNs and MPNNs use shared aggregation and update functions, which makes them naturally suited for homogenous graphs with nodes and edges of the same kind. But with specialized architectures like Heterogeneous Graph Neural Networks (HGNNs) [14], they can be extended to heterogeneous graphs, which have different types of nodes and edges. In order to manage a variety of relationships, these additions add type-specific parameters. On the other hand, because GBDTs ignore the graph structure and interpret inputs as separate feature vectors, they are not well suited for graph data. Due to this restriction, GBDTs are still a good option for conventional supervised learning tasks on tabular or feature-based datasets, while GCNs and MPNNs are better suited for tasks involving relational data [7]. "Graph Neural Network Contextual Embedding for Deep Learning on Tabular Data" demonstrates how GNNs can create contextual embeddings for tabular data. [15], which highlights the enhancement of feature representation through graph-based techniques.

Node embeddings produced by GCNs and MPNNs are extremely important because they encode both the properties of individual nodes and the structural context of their surroundings. These embeddings preserve relational information, allowing tasks such as node classification, link prediction, and graph classification to take advantage of the graph's underlying structure. GCNs and MPNNs have different ways of propagating information: GCNs employ a spectral method for homogeneous networks, while MPNNs generalize to heterogeneous graphs using customized message passing functions, making them more versatile [12,14] for complex relational data. It is advantageous to use both strategies, since they deal with distinct issues. GCNs are perfect for social networks or citation networks, since they are efficient and effective for semi-supervised tasks and homogeneous graphs [11]. On the other hand, due to their adaptable message passing architecture, MPNNs are excellent for managing dynamic and heterogeneous graphs, such as knowledge or molecular graphs [12].

According to Ivanov's [7] hybrid technique, "Boost Then Convolve: Gradient Boosting Meets Graph Neural Networks", gradient boosting models extract fea-

tures, which are then processed by GCNs. The complementary nature of graph-based and boosting learning is demonstrated by this two-step process. Through incorporation of domain knowledge, Kotelnikov et al. [16] present methods to improve the interpretability of GNNs for tabular data, making these models visible and efficient. To maximize efficiency on datasets with tabular node characteristics, for example, recent studies have suggested frameworks that combine GBDT and GNN [7,17,18].

Li et al. [19] offer a thorough taxonomy of GNN applications in tabular data in Graph Neural Networks for Tabular Data Learning: A Survey with Taxonomy & Directions. They also highlight important issues such as interpretability and scalability and promote hybrid models that combine GNNs with other machine learning methods. Modeling tabular data with diffusion models, also called TabDDPM [20], examines tabular data diffusion models and shows how specialized designs can improve predictive capabilities and capture feature dependencies.

Gradient boosting techniques such as LightGBM, XGBoost, and CatBoost are well known for their effectiveness, interpretability, and resilience. XGBoost is praised for its scalability and has been used to detect fraud in payment security systems by Zheng et al. [21]. The DCLGM model by Zhao et al. [22] demonstrates how combining LightGBM and deep learning overcomes the limitations of traditional recommendation systems by leveraging feature engineering and neural network capabilities. They consider LightGBM, as speed-optimized, to be useful in recommendation systems when combined with deep learning approaches.

Jang and Lee [23] proposed an innovative framework known as gradient-boosted graph neural networks (GBGNN). This model uniquely integrates several shallow Graph Neural Networks utilizing a gradient boosting strategy. In relation to node classification tasks, GBGNN shows considerable enhancements compared to conventional single GNN models. By binding the capabilities of Graph Convolutional Networks (GCNs), MPNNs, and gradient-boosting techniques, researchers are better equipped to address complex issues. Ultimately, this strategy signifies a promising path for creating more accurate and interpretable models within the field.

Our fusion strategy is based on classical ensemble learning techniques, namely stacked generalization (stacking), in which the outputs of many base learners are fused using a meta-learner to increase generalization performance. Wolpert [24] developed this idea, which was later backed up by foundational work such as bagging [25] and boosting [26]. Although our implementation employs modern gradient boosting models (CatBoost, XGBoost, and LightGBM), the ensemble structure is consistent with these early concepts.

3 Proposed Methodology

The proposed methodology is a four-step process. Graph embeddings or node embeddings are initially created using the GCN and the MPNN. The embeddings are then concatenated into a single feature vector. Individual predictions are then created using CatBoost, XGBoost, and LightGBM. A fusion model using logistic

regression combines these predictions for the final categorization. The conversion of an unprocessed dataset into a more organized and useful format is captured in Eq. (1).

$$Dataset_{processed} = f\,(Dataset) \tag{1}$$

here $f(Datasets)$ represents the function that transforms the dataset to the required format. In order to assure compatibility with machine learning methods, the function f converts class labels into numeric values, standardizes the data into a consistent format, and arranges the data into a tabular, flattened structure for effective analysis. In particular, f reads graph data from GXL files, extracts edge connections and node coordinates, labels the data according to class, and saves the processed data in CSV files. Figure 1 illustrates this process.

The graph $G = V, E$ is built with nodes $V = \{v_1, v_2, \ldots, v_N\}$ standing for entities and edges $E = \{e_{ij} \mid v_i, v_j \in V\}$ for relationships. Each node has a feature vector $\mathbf{x}_i \in \mathbf{R}^d$, and the weight $w_{ij} \in R$ of the edges indicates how strongly the links are held.

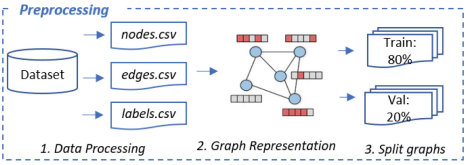


Fig. 1. Preprocessing

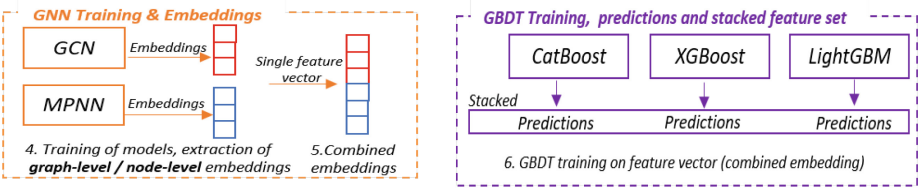
Node embeddings were computed using Eq. (2). The GCN and MPNN promulgate information through the graph to acquire node embeddings. For a given node v_i, the node embedding $h_i^{(l)}$ at layer l is updated using the following equation.

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{j \in N(i)} \frac{1}{c_{ij}} \mathbf{W}^{(l)} \mathbf{h}_\mathbf{j}^{(l)} + \mathbf{b}^{(l)}\right) \tag{2}$$

Where $N(i)$ is the set of neighbors of node i, c_{ij} is a normalization constant, $\mathbf{W}^{(l)}$ is the weight matrix at layer l, $\mathbf{b}^{(l)}$ is the bias term, and σ is an activation function. After multiple layers, the embedding of the node $\mathbf{h}_i^{(l)}$ is computed.

To create these embeddings, we train both the GCN and MPNN models in a supervised manner using class labels assigned to each graph. Specifically, both models are optimized jointly using a cross-entropy loss computed on the predicted graph representations. During training, node features are sent through two GNN layers before being aggregated into graph-level representations using global mean pooling. These embeddings are then refined using backpropagation over numerous epochs, with an early stopping based on validation loss. The retrieved embeddings from GCN and MPNN are later concatenated and fed into downstream ensemble classifiers. Section 4.2 provides further details.

In the next step, the embeddings generated from both GNN models are combined as explained in Fig. 2a. These acquired embeddings are used as features to train GBDT models in another stage to get their predictions as depicted in Fig. 2b. Suppose that y_i^{cat}, y_i^{xgb}, y_i^{lgb} are the base model predictions. They are stacked in a feature vector z_i for the proposed model $z_i = \left[y_i^{\text{cat}}, y_i^{\text{xgb}}, y_i^{\text{lgb}}\right]$. These models are trained using standard log-loss objectives (binary or multiclass depending on the dataset). This two-stage training strategy allows for successful embedding learning, followed by flexible downstream ensemble modeling.



(a) Computation of embeddings (b) Predictions and stacking

Fig. 2. Embeddings and prediction steps.

A logistic regression fusion model Eq. (3)is trained to produce the final result.

$$\hat{y}_i = \text{FusionModel}\,(z_i) \tag{3}$$

where y_i is the final predicted label for node i. The final evaluation uses standard classification metrics: accuracy and F1-scores (Fig. 3).

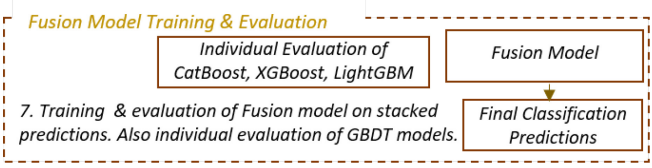


Fig. 3. Fusion training and evaluation.

4 Experiments

4.1 Datasets

The proposed methodology employs four datasets: Letters [32], Fingerprints [35], Freebase [30] and OGBG-MolHIV [31]. The Letters, Fingerprints and OGBG-MolHIV datasets are homogeneous and used for graph classification tasks [33]. They contain 6,750 graphs, 2,800 pictures and 41,127 graphs respectively. In comparison, the freebase dataset is heterogeneous and utilized for node

classification, with 180,098 nodes, 8 node types, and 36 edge types. The datasets include various network forms and complexities, allowing for evaluation of the methodology's performance across different graph architectures.

4.2 Implementation Details

The model was implemented in PyTorch-Geometric and ran on a personal laptop equipped with an Intel® Core™ i5-8250U CPU @ 1.60GHz with 8GB RAM and Intel® UHD Graphics 620 (integrated GPU), without using any dedicated GPU hardware. To provide transparency regarding computational requirements, we tracked and visualized the execution time of each pipeline component, including GNN training, embedding extraction, GBDT model training, and fusion. Given the modest hardware and runtime, the overall footprint of CO_2 of our experiments is minimal and aligns well with sustainable and reproducible research practices. The data sets stated in Sect. 4.1 were used to evaluate the model. To ensure a clean and structured format, the Pandas library parsed each.dat file and returned nodes, edges, and labels in csv format. These csv files were utilized to transform the data into a suitable graph structure (node features, edge and weights, and node labels) for GNN training. To maintain balanced class distributions, the nodes with labels were split into 80% training and 20% validation using stratified sampling.

GCN and MPNN architecture was implemented to learn node embeddings or graph embeddings. Both models were built using a two-layer GCN model using *GCNConv* layers. After the first layer, a *ReLU* activation function was utilized. Adam optimizer and cross-entropy loss were used for the training of both models with learning rate 0.001 for 1000 epochs. To avoid overfitting, early stopping was employed with a patience of 10, which was calculated using validation loss. The final embeddings for each node from both models were combined to form a single feature vector (low-dimensional representation embedding space).

As a base model, three machine learning models (CatBoost, XGBoost and LightGBM) were trained on feature vector (concatenated node embeddings). Logistic regression was used as a fusion model, which was trained on the predictions of the base models by creating a stacked ensemble. Finally, the performance of the proposed fusion model and individual base models was evaluated using the validation set with Accuracy: The percentage of correctly classified nodes. Micro-F1 Score: A weighted average of F1 scores in all classes. Macro-F1 Score: An unweighted average of F1 scores, treating all classes equally. The results of these metrics demonstrated that the proposed fusion model outperformed individual models as well as other models.

5 Results and Discussion

Table 1. Comparison of results

Dataset Type	Heterogeneous			Homogeneous														
Task	Node Classification			Graph Classification														
Dataset Name	Freebase			OGBG-MolHIV			Letters (Low)			Letters (Med)			Letters (High)			Fingerprints		
Models	Test Acc%	Micro F1	Macro F1	Test Acc%	Macro F1	Roc_ Auc	Test Acc%	Micro F1	Macro F1	Test Acc%	Micro F1	Macro F1	Test Acc%	Micro F1	Macro F1	Test Acc%	Micro F1	Macro F1
RpHGNN [27]	-	66.55	54.02	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SeHGNN [28]	-	63.41	50.71	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
GAT [29]	65.26	-	40.74	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SlotGAT [29]	-	66.83	49.68	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MPNN [5]	-	-	-	-	-	-	95.04	-	-	83.20	-	-	72.27	-	-	-	-	-
GNN local descriptor [5]	-	-	-	-	-	-	96.64	-	-	85.27	-	-	79.91	-	-	-	-	-
SoftHd [33]	-	-	-	-	-	-	95.56	-	-	85.35	-	-	71.75	-	-	-	-	-
DGNN using AAM loss [33]	-	-	-	-	-	-	98.87	-	-	90.57	-	-	83.34	-	-	-	-	-
GSN [34]	-	-	-	-	-	77.99	-	-	-	-	-		-	-	-	-	-	-
CatBoost	62.50	64.52	20.00	96.52	52.02	71.48	94.94	94.94	94.79	81.01	81.01	82.51	71.31	71.31	70.78	81.56	81.56	64.00
XGBoost	62.50	68.97	35.71	96.57	54.81	71.51	94.09	94.09	93.75	83.97	83.97	85.45	78.48	78.48	78.33	80.45	80.45	63.91
LightGBM	68.75	68.75	33.33	96.56	54.28	71.37	95.78	95.78	95.31	84.81	84.81	86.22	78.48	78.48	78.11	78.77	78.77	61.39
Fusion Model	68.75	73.33	40.45	89.10	56.61	67.56	96.62	96.62	96.28	84.81	84.81	86.88	78.06	78.06	77.66	80.45	80.45	62.46

Table 1 compares our proposed fusion model against existing GNN-based approaches across several benchmark datasets. The fusion model continuously performs well, beating individual GBDT models and several baseline GNNs in terms of accuracy, Micro-F1, and Macro-F1 scores. Notably, for the OGBG-MolHIV dataset, the fusion model yields a competitive ROC-AUC score of 67.56, demonstrating its robustness and generalizability across various graph topologies.

The Freebase dataset [30] contains 180,098 nodes, of which only 5,568 (3.09%) are labeled, highlighting a significant class imbalance. While the Micro-F1 metric emphasizes overall performance and is biased toward majority classes, the Macro-F1 metric evaluates per-class performance, treating all classes equally despite imbalance.

5.1 Ablation Study

We conducted an ablation study on homogeneous and heterogeneous datasets by removing specific components and evaluating their impact using standard metrics. Table 2 contains information on all scenarios and how they are implemented. As in scenario 1, two components (MPNN and Combined Embeddings) are omitted (×), but the other components (GCN, GCN Embeddings, CatBoost, XGBoost, and LightGBM) are included (✓). Each scenario is repeated five times and the standard deviation is calculated.

Its results are presented in Table 3 which also reflects higher values in bold. Table 3(a) shows the findings of an ablation study on a Freebase heterogeneous dataset for node classification using logistic regression. Three standard metrics discussed in Sect. 3 are assessed across nine distinct scenarios (1 through 9) in the Table 2. With Accuracy:68.75, Micro-F1: 73.33, and Macro-F1:40.45, Scenario-6 performs the best across all measures, making it the most efficient setup for

Table 2. Ablation study scenarios

Scenario	GCN	MPNN	GCN Embeddings	MPNN Embeddings	Combined Embeddings	Catboost	XGBoost	LightGBM
1	✓	×	✓	×	×	✓	✓	✓
2	×	✓	×	✓	×	✓	✓	✓
3	×	✓	×	✓	×	×	✓	✓
4	×	✓	×	✓	×	✓	×	✓
5	×	✓	×	✓	×	✓	✓	×
6	✓	✓	✓	✓	✓	✓	✓	×
7	✓	✓	✓	✓	✓	✓	×	✓
8	✓	✓	✓	✓	✓	×	✓	✓
9	✓	✓	✓	✓	✓	✓	✓	✓

heterogeneous freebase dataset. Values are much lower in worse performing scenarios (such as Scenarios 4 and 7), particularly for Micro-F1 and Macro-F1. In most situations, the accuracy stays above 0.6, with the exception of those that perform worse. In the majority of cases, the Micro-F1 and Macro-F1 scores are closer to one another, indicating balanced performance across all classes. Table 3(b) and (c) displays the final results for all ablation study scenarios on the Letters (Low) and OGBG-MolHIV data set respectively. From all of the scenarios, our proposed approach, which includes all components while studying ablation, has the highest value. So, we may say that all components contribute in the methodology.

Table 3. Ablation study results by excluding specific components

Dataset	(a) Freebase												(b) Letters (Low)				(c)OGBG-MolHIV	
Final Results	LR			Catboost			XGB			LGBM			LR	Cat Boost	XGB	LGBM	LR	
Scenario	Test Acc%	Micro -F1	Macro -F1	Test Acc%	Micro -F1	Macro -F1	Test Acc%	Micro -F1	Macro -F1	Test Acc%	Micro -F1	Macro -F1	Test Acc%	Test Acc%	Test Acc%	Test Acc%	Test Acc%	Roc_Auc
1	62.50	64.52	32.47	62.50	64.52	20.00	62.50	64.52	33.33	62.50	62.50	20.00	94.09	93.25	93.67	94.51	95.72	68.30
2	62.50	64.52	29.57	62.50	62.50	19.23	62.50	62.50	28.79	62.50	62.50	20.00	**96.62**	97.05	94.09	95.78	95.79	63.70
3	**68.75**	68.75	36.23	×	×	×	62.50	66.67	32.07	56.25	58.06	28.18	95.36	×	94.51	95.78	95.74	68.19
4	62.50	62.50	19.23	62.50	62.50	19.23	×	×	×	62.50	62.50	20.00	95.36	95.78	×	94.94	95.79	64.06
5	62.50	62.50	32.47	62.50	64.52	20.00	62.50	64.52	35.71	×	×	×	96.62	94.94	93.62	×	95.92	63.81
6	**68.75**	**73.33**	**40.45**	62.50	62.50	19.23	62.50	**68.97**	32.95	×	×	×	97.47	94.94	93.62	×	95.99	68.86
7	62.50	64.52	20.00	62.50	62.50	19.23	×	×	×	68.75	68.75	33.33	97.05	**94.94**	×	95.05	96.04	62.28
8	**68.75**	68.75	32.50	×	×	×	62.50	64.52	32.07	68.75	68.75	33.33	95.78	×	93.20	95.78	96.11	**73.30**
9	**68.75**	68.75	32.50	**62.50**	62.50	19.23	**62.50**	64.52	32.95	62.50	62.50	20.00	**96.62**	**94.94**	**94.09**	**95.78**	**96.25**	54.58

6 Conclusion and Future Work

We introduce a new GNN and GBDT fusion framework for graph and node classification, which effectively unifies the model and increases it in a mutually enhancing manner for a resilient classification task. They extract graph-level embeddings and node-level embeddings, respectively, for complex structural and

relational features of the graph data by using GCN and MPNN. This matrix of embeddings is then joined into an integrated feature vector to increase its representative capacity. Gradient-boosting decision tree models (e.g., CatBoost, XGBoost, LightGBM) make predictions on their own using these embeddings, taking advantage of their ability to handle complex interactions between features. A logistic regression based fusion model finally combines these predictions to give a fused and accurate classification output. This integrated approach showcases how powerful it can be to combine a graph-based approach with a machine learning model for a node classification or network classification task.

Future Work. Exploring more flexible and supplementary fusion approaches for the scalability of larger and more complicated datasets. Fine-tune this model for subgraph spotting. The generation of a novel heterogeneous graph classification dataset.

Acknowledgments. This work was partially supported by the "PHC PERIDOT" program (project number: 51405PL), funded by the French Ministry for Europe and Foreign Affairs, the French Ministry for Higher Education and Research and Higher Education Commission (HEC) in Pakistan.

Disclosure of Interests. Authors declare no conflict of interests.

References

1. Sun, Y., Han, J.: Mining heterogeneous information networks: principles and methodologies. Morgan & Claypool (2012)
2. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
3. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
4. Kajla, N.I., Missen, M., Coustaty, M., Badar, H., Pasha, M., Belbachir, F.: A histogram-based approach to calculate graph similarity using graph neural networks. Pattern Recognit. Lett. **186**, 286–291 (2024)
5. Kajla, N.I., Missen, M., Luqman, M.M., Coustaty, M.: Graph neural networks using local descriptions in attributed graphs: an application to symbol recognition and handwritten character recognition. IEEE Access **9**, 99103–99111 (2021)
6. Kaur, H., Nori, H., Jenkins, S., Caruana, R., Wallach, H., Vaughan, J.W.: Interpreting interpretability: understanding data scientists' use of interpretability tools for machine learning. In: Proceedings of CHI Conference on Human Factors in Computing Systems, pp. 1–14 (2020)
7. Ivanov, S., Prokhorenkova, L.: Boost then convolve: gradient boosting meets graph neural networks. arXiv preprint arXiv:2101.08543 (2021)
8. Chen, J., et al.: Does your graph need a confidence boost? Convergent boosted smoothing on graphs with tabular node features. arXiv preprint arXiv:2109.06378 (2021)
9. Hong, M.-Y., Chang, S.-Y., Hsu, H.-W., Huang, Y.-H., Wang, C.-Y., Lin, C.: TreeXGNN: can gradient-boosted decision trees help boost heterogeneous graph neural networks? In: ICASSP 2023 - IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 1–5 (2023)

10. Bazhenov, G., Platonov, O., Prokhorenkova, L.: TabGraphs: A benchmark and strong baselines for learning on graphs with tabular node features. arXiv preprint arXiv:2409.14500 (2024)
11. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations (ICLR) (2017)
12. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: International Conference on Machine Learning (ICML) (2017)
13. Chen, T., Guestrin, C.: XGBoost: a scalable tree boosting system. In: Proceedings of 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD) (2016)
14. Schlichtkrull, M., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: European Semantic Web Conference (ESWC) (2018)
15. Villaizán-Vallelado, M., Salvatori, M., Carro, B., Sanchez-Esguevillas, A.J.: Graph neural network contextual embedding for deep learning on tabular data. Neural Netw. **173**, 106180 (2024)
16. Alkhatib, A., Boström, H.: Interpretable graph neural networks for heterogeneous tabular data. arXiv preprint arXiv:2408.07661 (2024)
17. Nath, P., Waghmare, G., Agrawal, N., Kumar, N., Asthana, S.: TBoost: gradient boosting temporal graph neural networks. In: Advances in Neural Information Processing Systems (NeurIPS) (2023)
18. Ye, H., Xu, H., Wang, H., Wang, C., Jiang, Y.: GNN&GBDT-guided fast optimizing framework for large-scale integer programming. In: International Conference on Machine Learning (ICML), pp. 39864–39878 (2023)
19. Li, C.-T., Tsai, Y.-C., Liao, J.C.: Graph neural networks for tabular data learning. In: Proceedings of IEEE International Conference on Data Engineering (ICDE), pp. 3589–3592 (2023)
20. Kotelnikov, A., Baranchuk, D., Rubachev, I., Babenko, A.: TabDDPM: modelling tabular data with diffusion models. In: International Conference on Machine Learning (ICML), pp. 17564–17579 (2023)
21. Zheng, Q., Yu, C., Cao, J., Xu, Y., Xing, Q., Jin, Y.: Advanced payment security system: XGBoost, CatBoost and SMOTE integrated. arXiv preprint arXiv:2406.04658 (2024)
22. Zhao, B., Li, B., Zhang, J., Cao, W., Gao, Y.: DCLGM: fusion recommendation model based on LightGBM and deep learning. Neural Process. Lett. **56**(1), 17 (2024)
23. Jang, E., Lee, K.Y.: GBGNN: gradient boosted graph neural networks. J. Inf. Process. Syst. **20**(4), 501–513 (2024)
24. Wolpert, D.H.: Stacked generalization. Neural Netw. **5**(2), 241–259 (1992)
25. Breiman, L.: Bagging predictors. Mach. Learn. **24**(2), 123–140 (1996)
26. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. **55**(1), 119–139 (1997)
27. Efficient Heterogeneous Graph Learning via Link Prediction. https://paperswithcode.com/paper/efficient-heterogeneous-graph-learning-via
28. Simple and Efficient Heterogeneous Graph Neural Network. https://paperswithcode.com/paper/simple-and-efficient-heterogeneous-graph
29. Heterogeneous Node Classification on Freebase (Heterogeneous Node Classification). https://paperswithcode.com/sota/heterogeneous-node-classification-on-freebase

30. THUDM, Heterogeneous Graph Benchmark (HGB) (2022). https://github.com/thudm/hgb. Accessed Dec 2024
31. Hu, W., et al.: Open graph benchmark: datasets for machine learning on graphs. Adv. Neural. Inf. Process. Syst. **33**, 22118–22133 (2020)
32. Riesen, K., Bunke, H.: IAM graph database repository for graph-based pattern recognition and machine learning. In: SSPR & SPR, pp. 287–297. Springer (2008)
33. Kajla, N.I., Missen, M., Luqman, M.M., Coustaty, M., Mehmood, A., Choi, G.S.: Additive angular margin loss in deep graph neural network classifier for learning graph edit distance. IEEE Access **8**, 201752–201761 (2020)
34. Bouritsas, G., Frasca, F., Zafeiriou, S., Bronstein, M.M.: Improving graph neural network expressivity via subgraph isomorphism counting. IEEE Trans. Pattern Anal. Mach. Intell. **45**(1), 657–668 (2022)
35. Watson, C.I., Wilson, C.L.: NIST Special Database 4: Fingerprint Database. National Institute of Standards and Technology (1992)