Journal of Internet Services and Applications, 2026, 17:1, doi: 10.5753/jisa.2026.7501

A Decoupled Embedding-Based Framework for Efficient Node Classification on Social Networks

Erick S. Nascimento [**State University of Ceará** | ***erick.nascimento@aluno.uece.br***]
Ivo A. Pimenta [**State University of Ceará** | ***aguiar.pimenta@aluno.uece.br***]
Marcello H. Lee [**State University of Ceará** | ***marcello.lee@aluno.uece.br***]
Thelmo de Araujo [**State University of Ceará** | ***thelmo.araujo@uece.br***]
Rafael L. Gomes [**State University of Ceará** | ***rafa.lopes@uece.br***]

Center of Science and Technology, State University of Ceará, Fortaleza, CE, 60714-903, Brazil.

Received: 04 December 2025 • **Accepted:** 01 April 2026 • **Published:** 29 May 2026

Abstract Graph-based learning has become a cornerstone for analyzing complex relationships in domains such as social networks. However, most existing solutions rely on Graph Neural Networks (GNNs), which often require high computational resources, careful parameter tuning, and extensive training time. In this paper, we propose a decoupled learning framework for node classification that replaces end-to-end deep graph architectures with latent structural embeddings and conventional machine-learning models. Our method generates low-dimensional node embeddings and then applies classifiers such as logistic regression, KNN, and random forests to predict node categories. By structurally decoupling the embedding and classification stages, our approach achieves competitive performance while drastically reducing memory complexity and training time. Experimental results on multiple social network datasets demonstrate that our method offers a scalable, interpretable alternative to deep GNN models.

Keywords: Embedding, Graph Representation, Social Network, Machine Learning.

1 Introduction

Graphs are powerful and versatile data structures capable of modeling entities and the relationships between them in a wide range of domains, including social networks, biological systems, citation networks, and financial transactions [Khoshraftar and An, 2024]. The ability to capture complex relationships between nodes makes graphs fundamental for tasks such as node classification [Wang *et al*., 2017], link prediction [Wei *et al*., 2017], and community detection [Fortunato, 2010]. However, these tasks traditionally require machine learning techniques capable of handling the high dimensionality and topological complexity of relational data.

The advancement of graph embedding techniques has emerged as an efficient solution to this challenge. Graph embedding converts graph data into a low-dimensional vector space, preserving the original structural properties as much as possible [Cai *et al*., 2018]. This transformation allows traditional supervised or unsupervised learning algorithms to operate on the resulting vectors, avoiding the computational costs associated with directly processing large adjacency matrices. Graph embedding provides an effective yet efficient way to solve graph analytics problems by transforming the graph into a low-dimensional space where the graph information is preserved [Cai *et al*., 2018].

From a conceptual standpoint, Graph Representation Learning (GRL) can be divided into two broad categories: traditional embedding methods and methods based on graph neural networks (GNNs) [Khoshraftar and An, 2024].The former utilize approaches such as matrix factorization [Brochier *et al*., 2019; Epasto and Perozzi, 2019; Zhu and Koniusz, 2021], random walks [Heidari and Papagelis, 2020; Xiao *et al*., 2020], and autoencoders [Goyal *et al*., 2018; Mahdavi *et al*., 2019; Wang *et al*., 2016], while the latter employ hierarchical neighborhood aggregation and convolutional operations over the graph structure.

In this context, it is important to highlight that embeddings allow for the construction of vector spaces for nodes, edges, and subgraphs. This process is inspired by the idea of adapting computational allocation to minimize distortion while maintaining utility [Pimenta *et al*., 2025, 2024; Silveira *et al*., 2022; Costa *et al*., 2020] and preserving internal graph properties so that learning tasks can be solved by conventional vector-based frameworks [Makarov *et al*., 2021]. That is, once the graph is represented in a compact vector format, it is possible to apply classic models, such as logistic regression, random forests, or support vector machines, without the need for deep architectures. This approach drastically reduces training time and memory consumption while maintaining competitive performance on structural tasks.

Embedding techniques vary according to the type of graph and the desired output granularity: node embeddings to represent individual nodes, edge embeddings for pairs of nodes, or whole-graph embeddings for graph classification tasks [Khoshraftar and An, 2024; Cai *et al*., 2018]. Regardless of the approach, the central objective is to preserve structural and semantic properties—such as proximity, structural equivalence, and node roles (role equivalence), which are essential for structural node classification, meaning the identification of similar topological roles even in distinct regions of the graph.

Therefore, this work proposes an alternative approach to graph neural networks, exploring the use of structural embeddings to transform graph datasets of varying sizes and

topologies into vector representations compatible with classic machine learning models. Through this methodology, it is possible to address structural node classification tasks with greater computational efficiency, without significant loss of performance, offering a practical solution for large-scale applications or on devices with limited resources.

Experimental results on the MUSAE-Facebook and MUSAE-GitHub datasets reveal a deliberate trade-off between predictive performance and computational efficiency. While the proposed decoupled framework observes a marginal accuracy drop of approximately 4% in the Facebook dataset compared to end-to-end baselines (90% vs. 94%), it achieves performance parity in the GitHub dataset ($\sim$85%). Crucially, this trade-off yields substantial gains in resource efficiency, reducing peak memory consumption by up to $10\times$ (dropping from over 11 GB in GAT baselines to roughly 1.2 GB) and accelerating the training pipeline from thousands of seconds to less than a minute. These findings validate the approach as a scalable alternative for resource-constrained environments where monolithic GNNs can be prohibitive.

The main contribution of this work is the proposal and evaluation of a modular, decoupled learning framework that treats graph representation and node classification as distinct stages. By replacing monolithic end-to-end GNN training with a pipeline composed of an unsupervised neural encoder (VGAE) followed by lightweight traditional classifiers, we demonstrate how to overcome the memory bottlenecks inherent to standard message-passing architectures ($\mathcal{O}(N \times F)$). This design enables a "train once, classify everywhere" paradigm, where a single structural embedding can support multiple downstream tasks with minimal computational overhead.

The remainder of this work is organized as follows. Section 2 presents the main related studies on graph representation learning and structural node classification. Section 3 details the proposed lightweight framework, describing the embedding generation process and the classification pipeline. Section 4 describes the datasets, metrics, and experimental setup employed to validate our methodology. Section 5 discusses the obtained performance results in terms of effectiveness and computational efficiency. Lastly, Section 6 summarizes the main contributions of the work and outlines future research directions.

2 Related Work

This section describes the main works related to network management and performance analysis published recently, considering performance and quality issues. Table 1 presents several existing proposals (column *Ref.*), where the *Description* column represents the strategy of the corresponding proposal, while column *Contribution* highlights positive points of it.

Luo *et al*. [2024] present a novel graph-tokenization framework that generates structure-aware, semantic node identifiers (IDs) in the form of short sequences of discrete codes. These node IDs are derived via vector quantization on node embeddings from multiple layers of a GNN, thereby compressing continuous representations into compact symbolic codes that still reflect multi-order neighborhood structure. Experiments across 34 datasets demonstrate that these IDs not only improve inference efficiency but also achieve competitive performance compared to state-of-the-art methods.

Li *et al*. [2024] introduce a framework for embedding nodes in property graphs by way of latent neighborhood sampling. Their method emphasizes how to select and evaluate latent neighborhoods for better node classification performance. The contribution is two-fold: first, tailoring embeddings to property graphs; second, defining a quantitative metric for the usefulness of a node's neighborhood in classification tasks. Moreover, by incorporating node attributes directly into the embedding process, the framework enables more discriminative representations in heterogeneous graph structures, especially when relational context alone is insufficient.

Dalvi and Honavar [2025] propose mapping node features into very high-dimensional hypervectors and aggregating neighborhood information through hyperdimensional operators. Their "one-pass" learning algorithm serves as an alternative to traditional GNNs with lower computational cost while maintaining competitive accuracy on node classification and link prediction. This approach demonstrates the potential of algebraic operations in high-dimensional vector spaces to encode rich relational patterns without iterative message passing, thus reducing both training time and model complexity.

Zhao *et al*. [2024] propose a probing framework to investigate what kinds of graph knowledge are encoded in learned embeddings. The key contribution is providing systematic evaluation of embedding methods' capabilities and limitations, thus guiding the design of embedding techniques tailored to specific downstream tasks. Their results highlight that different embedding strategies tend to prioritize distinct structural cues, revealing the importance of selecting appropriate embedding properties aligned with task-specific requirements.

Lecca and Lecca [2023] explore applications of graph embedding and geometric deep learning in domains like network biology and structural chemistry. Among its insights is the observation that embeddings enable simpler models (e.g., SVMs or logistic regression) to operate effectively on large graph-structured data, whereas deep models may become computationally prohibitive. Their analysis reinforces the relevance of scalable solutions, particularly in domains where datasets grow rapidly and computational resources may be limited.

Khoshraftar and An [2024] provide a comprehensive review of both traditional embedding methods and GNN-based techniques, covering static and dynamic graphs. Their contribution is to articulate the practical limitations of GNNs, such as scalability issues, over-smoothing, and memory demands, and to highlight embedding methods as viable, efficient alternatives. The survey also categorizes key research trends, showing a progressive shift toward hybrid techniques that combine structural preservation with computational affordability.

Chen *et al*. [2023] design a graph-learning block that simultaneously models structural and semantic aspects of dynamic graphs (temporal dimension included). Their primary contribution is showing that embedding methods that account for both structure and semantics, as well as temporal evolution,

Table 1. Related Work

Ref.	Description	Contribution
Luo *et al*. [2024]	Proposes a graph-tokenization framework that generates structure-aware, semantic node IDs (discrete codes) as compact symbolic node representations.	Introduces compact, interpretable node IDs that increase inference efficiency and maintain competitive performance across 34 datasets
Li *et al*. [2024]	Introduces the EPGE (Enhanced Property Graph Embedding) framework for embedding nodes in property graphs (nodes + attributes) by latent neighborhood sampling	Improves node classification in property graphs by sampling latent neighborhoods and defining a quantitative metric for neighborhood usefulness.
Dalvi and Honavar [2025]	Proposes a hyperdimensional representation learning method (HDGL) mapping node features to very high dimensional hypervectors and aggregating via HD operators.	Offers a one-pass learning algorithm alternative to GNNs, competitive in accuracy and lower in computational cost, for node classification/link prediction.
Zhao *et al*. [2024]	Presents a framework to probe which types of graph knowledge (structural, path-wise, etc) are encoded in learned embeddings.	Provides systematic evaluation of embedding capabilities and limitations, guiding design of future embedding methods.
Lecca and Lecca [2023]	A survey reviewing graph embedding and geometric deep learning in biology/chemical networks, highlighting embedding methods in complex domains.	Emphasizes shift from heavy deep models to efficient vector representations enabling simpler models (e.g., SVM) on large graph-structured data.
Wang *et al*. [2025]	Comprehensive review of graph embedding (traditional) and GNN-based methods for static and dynamic graphs, summarizing taxonomy and limitations.	Articulates limitations of GNNs (scalability, over-squashing) and outlines future directions for GRL.
Lin *et al*. [2023]	Uses dual attention mechanism (structure + features) for node classification in graphs, focusing on structural role embedding.	Demonstrates improved node classification by integrating structural attention, reinforcing the importance of structural roles and embeddings for nodes.
Khan *et al*. [2025]	Introduces the Distance-Driven Graph Neural Network (DDGNN), a novel architecture that integrates distance-aware features and marker nodes to enhance structural representation in node classification tasks.	Improves structural differentiation and alleviates the over-smoothing problem in deep GNNs by encoding inter-node distances, achieving higher classification accuracy on complex graph topologies.
Our Proposal	Generation node embeddings followed by standard ML classifiers, enabling efficient structural node classification without GNNs.	Enables efficient structural node classification by transforming graphs into fixed-size vector representations and applying lightweight *sklearn* models.

can lead to robust node representations in evolving graphs. This demonstrates that capturing temporal dependencies is essential for real-world networks, where the graph topology and attributes frequently change over time.

Lin *et al*. [2023] propose a model that integrates two attention branches: an internode representation encoding that captures relationships between nodes (structure) and an intranode contextual reasoning branch that captures node-internal attribute context. Their evaluation across ten datasets shows that integrating both structural and attribute-based attention improves node classification performance.

Khan *et al*. [2025] introduced the Distance-Driven Graph Neural Network (DDGNN), a model designed to enhance structural node classification by explicitly incorporating distance-aware features into the graph learning process. Unlike conventional GNNs that rely solely on local neighborhood aggregation, DDGNN employs marker nodes to encode inter-node distances, thus capturing both local and global structural dependencies. This mechanism allows the model to better differentiate nodes with similar local contexts but distinct topological roles, effectively mitigating the over-smoothing issue commonly observed in deeper architectures. Experimental results on multiple public graph datasets demonstrated that DDGNN achieves improved classification accuracy and robustness across complex graph structures. While the approach still depends on message passing and deep parameterization, it reinforces the importance of structure-preserving representations in node classification—an idea also central to our proposed lightweight neural embedding framework.

In this work, unlike most existing methods that rely on deep graph neural networks or highly complex embedding pipelines, we propose a lightweight yet effective approach for structural node classification that departs from heavy GNN architectures. Once these embeddings are obtained, they are fed as fixed-size vectors into standard machine-learning classifiers from the scikit-learn toolkit. Unlike prior works, which often integrate embedding and classification within a single deep architecture, none explicitly decouples these stages to balance structural expressiveness with computational efficiency. Our methodology introduces this separation, preserving topological information through neural structural embeddings while enabling the use of lightweight, interpretable classifiers. This design allows handling graphs of varying sizes and topologies without the typical memory and training overhead associated with deep GNN models.

3 Proposal

In this section, we present our proposed framework for structural node classification in social-network graphs. The motivation behind our approach stems from the need to reduce the computational complexity of Graph Neural Networks (GNNs) while preserving structural information that is crucial for understanding node roles in social topologies. Instead of relying on deep message-passing architectures, we adopt a lightweight embedding-based methodology that transforms social graphs into compact vector representations suitable for traditional machine learning models.

3.1 Overview

The proposed pipeline, illustrated in Figure 1, follows a three-stage process. First, a social network dataset is represented as a graph where nodes correspond to users and edges denote social relationships (e.g., friendship or follower links). Second, an embedding method is applied to encode structural patterns of the network into a low-dimensional representation. Finally, these embeddings are used as input features for conventional machine-learning classifiers to perform structural node classification, identifying nodes that share similar topological roles.

The proposed framework operates through a streamlined pipeline designed to decouple **graph representation learning** from task-specific classification.

First, Latent Embedding Generation: The raw graph data, comprising the adjacency matrix and node features, is fed into a Variational Graph Autoencoder (VGAE). This neural encoder is trained in an unsupervised manner to optimize a link reconstruction loss, learning to compress high-dimensional sparse graph data into compact, low-dimensional latent vectors ($\mathbf{z}_i$).

Second, Feature Extraction and Classification: Once the encoder converges (or acts as a randomized projector after a single epoch), it functions as a static feature extractor, transforming all nodes into a dense embedding matrix. These generated embeddings then serve as the input feature space for standard supervised machine learning algorithms (e.g., XGBoost, Random Forest). This separation ensures that the computationally expensive message-passing operations are isolated in the first stage, allowing for rapid retraining and inference in the classification stage without re-processing the graph topology.

3.2 Structural Embedding Generation

The embedding generation phase constitutes the core of our proposed approach. Rather than employing handcrafted structural descriptors or directly a message-passing Graph Neural Networks (GNNs), we adopt a purely neural architecture to learn representations directly from the graph. It is important to emphasize that our embedding generation model never has access to the labels, learning in a purely unsupervised manner. The goal is to obtain compact, low-dimensional vectors that preserve the characteristics of the social network, enabling efficient downstream node classification.

Graph representation. We consider a social network graph defined as $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of users and $E \subseteq V \times V$ represents social relationships (friendship, follow, or interaction links). Each node v_i is encoded using an adjacency-based representation that captures its local connectivity pattern within G. This representation serves as the input to the neural embedding model.

To map each node into a continuous latent space, the **neural embedding model** we employ is a Variational Graph Autoencoder (VGAE), a self-supervised GNN, trained in an unsupervised manner to reconstruct or predict relations. Formally, the network learns a mapping function

$$f_\theta : v_i \in V \rightarrow \mathbf{z}_i \in \mathbb{R}^d ,$$

where $\mathbf{z}_i$ is the embedding vector with dimension $d \in \{32, 64, 128\}$. The network parameters θ are optimized to minimize the reconstruction loss between the observed adjacency matrix A and the predicted one $\hat{A}$, expressed as

$$\mathcal{L}_{rec} = \|A - \hat{A}\|_2^2 ,$$

where $\hat{A} = \sigma(ZZ^T)$, Z is the matrix of learned embeddings and $\sigma(\cdot)$ is a sigmoid activation function. This formulation is analogous to a neural autoencoder operating on graph connectivity patterns, capturing both local and global structural information through the optimization process.

Embedding dimensionality plays a crucial role in determining the trade-off between representational capacity and computational efficiency. In our framework, the dimensionality of the latent space (d) can be adjusted to better reflect the structure and scale of the social network under analysis. Smaller values of d tend to yield more lightweight embeddings, resulting in faster training and lower memory usage, whereas larger values are capable of preserving more subtle and global structural patterns present in complex topologies. This configurability allows the model to adapt to heterogeneous datasets and constraints, supporting applications that range from resource-limited environments to high-fidelity structural learning.

After the embedding process has converged, or even with minimal training, the final representation of each node is stored in a dense matrix of shape $O(N \times D)$, where N is the number of nodes and D is the dimensionality of the learned embedding. This setup follows the standard representation-learning paradigm, in which nodes (or tokens) are mapped to compact latent vectors, an idea that demonstrates the effectiveness of fixed-dimensional embeddings such as 300 or 1000 dimensions Mikolov *et al*. [2013]. In contrast, conventional GNNs operate directly on the original feature matrix $X \in \mathbb{R}^{N \times F}$, where F denotes the number of raw input features per node, and each propagation layer produces hidden states $H^{(l)} \in \mathbb{R}^{N \times D_{\text{hidden}}}$, with D_{hidden} representing the dimensionality of the internal GNN layers Kipf [2016]. Here, N specifies how many nodes compose the dataset or graph, F defines the size of the input feature vector associated with each node, and D corresponds to the size of the compact latent representation produced by the embedding process—highlighting how our approach reduces the computational footprint from the original $N \times F$ space to the more efficient $N \times D$ representation.

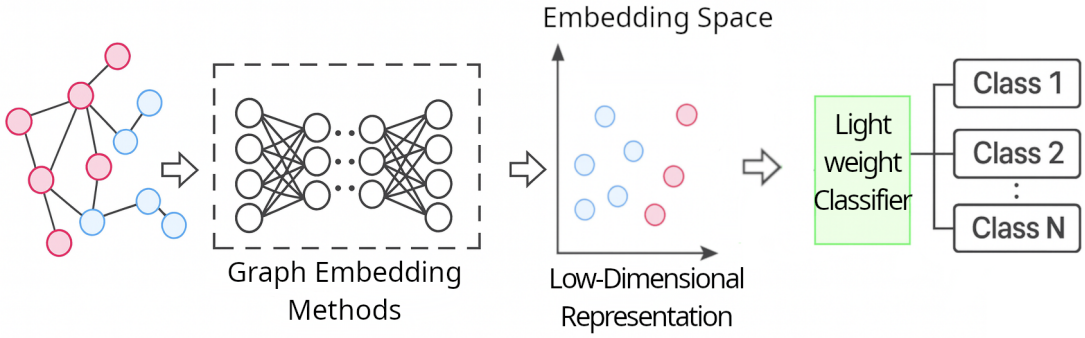

Figure 1. Proposal Overview.

As a result, our decoupled approach reduces memory usage from the $O(N \times F)$ requirement of GNNs to the more compact $O(N \times D)$ embedding space and avoids the repeated $O(N \times D_{\text{hidden}})$ message-passing operations characteristic of GNN training, ensuring significantly improved scalability under hardware constraints.

Once the embedding process is complete, each node v_i is represented by its learned embedding vector $\mathbf{z}_i$. These vectors are subsequently used as feature inputs to traditional machine-learning classifiers for structural node classification, ensuring scalability to large social network graphs while maintaining low memory and runtime overhead.

3.3 Feature Preparation and Classification

Once the embedding model has converged, each node $v_i \in V$ is represented by its corresponding latent vector $\mathbf{z}_i \in \mathbb{R}^d$. These embeddings constitute a compact feature space that encodes the topological characteristics of the social graph in a form suitable for downstream learning tasks. Because the embeddings are fixed-length and independent of recursive message passing, they can be directly integrated with standard machine-learning models, enabling an efficient and transparent classification pipeline.

Before classification, all embedding vectors are normalized using ℓ_2 scaling to mitigate magnitude disparities across dimensions and to stabilize the optimization process of the classifiers. No further dimensionality reduction or feature engineering is required, since the embedding process itself performs a structural abstraction of the original graph. This design choice allows the entire framework to remain lightweight and easily reproducible.

The prepared embeddings are used as input to conventional classifiers implemented in the scikit-learn library, including logistic regression, random forest models, and others. These algorithms were selected for their balance between computational simplicity and robustness in handling classification tasks. Logistic Regression and Random Forest are computationally lighter than methods like Graph Neural Networks (GNNs), offering significantly faster training and inference times when applied to pre-calculated vector embeddings. Their straightforward and optimized implementation within the scikit-learn library Pedregosa *et al*. [2011] further ensures comparability and computational efficiency across the models. Each classifier is trained independently on the same embedding representations, allowing a direct comparison of model behavior and efficiency across different learning paradigms.

In summary, the proposed framework establishes an efficient and interpretable pipeline for structural node classification in social-network graphs. By combining neural-based embeddings with traditional machine-learning classifiers, the model preserves the structural expressiveness of graph data while minimizing computational cost and implementation complexity. The modular design also enables flexible experimentation across different embedding configurations and classification algorithms.

The following section presents the experimental evaluation, where we assess the proposed framework using real-world social-network datasets. We detail the embedding settings, classification models, and evaluation metrics employed, followed by a discussion of the obtained results in terms of performance and scalability.

4 Experiments

This section presents the experimental evaluation conducted to validate the effectiveness and efficiency of the proposed framework for structural node classification in social-network graphs. The experiments were designed to assess how well the neural-based embeddings, when combined with traditional machine-learning classifiers, preserve topological characteristics and achieve competitive predictive performance compared to Graph Neural Networks (GNNs), while maintaining significantly lower computational cost. We describe the datasets used, the experimental setup, the evaluation metrics, and the performance results obtained across different configurations of embedding dimensionality and classifiers. The evaluation aims to answer three core questions: (i) How effective are the learned embeddings for node classification? (ii) How does the framework scale across networks with distinct topological patterns? (iii) What is the trade-off between predictive accuracy and computational efficiency?

The experiments were implemented in Python, using Py-

Torch and PyTorch Geometric for the development of the GNN models (VGAE, GCN, GAT). The lightweight classifiers were implemented with scikit-learn and XGBoost. Memory monitoring was performed using the *psutil* and *memory_profiler* libraries, and the measurement of execution time (training and inference) was obtained using the `time.process_time()` function from Python's *time* module.

4.1 Datasets

To evaluate the proposed framework, we employ two real-world social-network datasets from the MUSAE (Multi-Scale Attributed Node Embedding) collection: **MUSAE-Facebook** (hereafter referred to as **D1**) Rozemberczki *et al*. [2019] and **MUSAE-GitHub** (hereafter referred to as **D2**) Rozemberczki *et al*. [2019]. and are widely adopted benchmarks for structural node classification, link prediction, and community detection tasks. These datasets provide complementary graph structures and attribute distributions, allowing assessment of the framework's generalization across heterogeneous social domains.

The **D1** (MUSAE-Facebook) dataset represents a page–by–page webgraph of verified Facebook sites, where nodes correspond to official Facebook pages and edges denote mutual "likes" between them. Node features are textual embeddings extracted from each page's description field, summarizing its purpose. The dataset was collected via the Facebook Graph API (November 2017) and filtered to include pages from four major categories: politicians, governmental organizations, television shows, and companies. The experimental task associated with this graph is multi-class node classification across these four categories.

The **D2** (GitHub) network consists of developers as nodes and follower relationships as edges. The MUSAE-GitHub dataset was collected from the GitHub public API (June 2019), including only users who have starred at least ten repositories. Node features encode profile-based attributes such as location, starred repositories, employer, and email domain. The target variable is binary, distinguishing web developers from machine-learning developers, derived from users' job titles.

4.2 Evaluation Metrics

To comprehensively evaluate the performance of the proposed framework, we employ a set of quantitative metrics that jointly capture predictive accuracy, class balance robustness, and computational efficiency.

Accuracy measures the proportion of correctly classified nodes over the total number of nodes, as represented in Equation 1:

$$\mathrm{ACC} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}(\hat{y}_i = y_i)\,, \tag{1}$$

where $\mathbf{1}(\cdot)$ is the indicator function, $\hat{y}_i$ and y_i denote the predicted and true labels for node i, respectively. In the same way, the weighted F1-score accounts for class imbalance by weighting the contribution of each class according to its support (i.e., the number of samples belonging to that class). Thus, more frequent classes have a greater influence on the final metric. It is given by Equation 2:

$$\mathrm{F1}_{weighted} = \sum_{c=1}^{C} \frac{n_c}{N} \times \frac{2 \times \mathrm{Precision}_c \times \mathrm{Recall}_c}{\mathrm{Precision}_c + \mathrm{Recall}_c}\,, \tag{2}$$

where n_c is the number of samples in class c, N is the total number of samples, $\mathrm{Precision}_c$ and Recall_c are the Precision and the Recall of class c, respectively.

Training time records the total time required for model training and embedding generation, reflecting computational efficiency. Similarly, **Memory usage** measures the peak RAM consumption during the training process, emphasizing the lightweight nature of the framework compared to GNN-based baselines.

These metrics collectively allow evaluation from both *utility* (classification quality) and *efficiency* (time and resource consumption) perspectives, aligning with the framework's dual objective of achieving competitive accuracy with reduced computational overhead.

4.3 Embedding Configuration

The embedding model was configured to generate fixed-length vector representations for each node while ensuring a favorable trade-off between representational capacity and computational cost. To assess the impact of embedding dimensionality on classification performance, we considered three distinct sizes: $d \in \{32, 64, 128\}$. Smaller values of d promote compactness and lower memory usage, whereas larger embeddings may encode richer attribute information at the cost of increased resource consumption.

All embeddings were trained in an unsupervised manner for a single epoch, as initial ablation experiments showed no significant performance gain from prolonged training. This setting minimizes execution time while preserving competitive predictive quality on downstream tasks. Once trained, the embeddings remained frozen and were used as the sole input features for all evaluated classifiers, ensuring a fair comparison across models and datasets.

The computational efficiency of this design stems from its linear complexity with respect to the number of nodes, i.e., $\mathcal{O}(|V| \cdot d)$, which compares favorably against the $\mathcal{O}(|V| \cdot F)$ footprint of multi-hot feature encodings commonly required by Graph Neural Networks (GNNs). As demonstrated in our experiments, this leads to substantial reductions in both memory usage and training time while maintaining competitive performance in structural node classification tasks on real-world social graphs.

5 Results

Our experiments compare our **Approach** (VGAE $\mathcal{O}(N \times D)$ + Lightweight Classifier) with the **Baseline Approach** (GCN/GAT $\mathcal{O}(N \times F)$ across a series of metrics that we will explore in this section. The goal here is to show competitive results even with much lower memory complexity. We tested embeddings of sizes 32, 64, and 128 across various classification models. Additionally, we tested 1 and 200 training

epochs in the embedding generator, with the number of training epochs indicated in parentheses. The values shown here refer to the test samples.

This section presents the evaluation results of our lightweight framework compared to the GCN and GAT baselines. To ensure a fair and comprehensive analysis, we assess four key aspects: (i) node classification accuracy; (ii) robustness to class imbalance through Weighted F1-Score; (iii) execution efficiency measured by Training Time; and (iv) computational feasibility based on Memory Usage. All evaluations were performed using embeddings with dimensions $d \in \{32, 64, 128\}$, combined with multiple classifiers (MLP, KNN, LR, RF, XGB), while the GNN baselines were tested using their standard training configurations. The best performing classifier for each setting is highlighted in the corresponding tables to support direct comparison.

5.1 Classification Results

Table 2 presents the accuracy results obtained for the Musae Facebook (D1) and Musae GitHub (D2) datasets in all all evaluated configurations, enabling a comprehensive analysis of how the performance of the model scales with the dimensionality of the embedding, as well as how the embedding-based methods compare against other GNN architectures.

Table 2. Classification Performance Comparison (Accuracy)

Data	E	MLP	KNN	LR	RF	XGB	GCN	GAT
D1	32	0.78	0.88	0.68	0.85	0.84	-	-
D1	64	0.83	0.90	0.73	0.89	0.88	-	-
D1	128	0.86	0.90	0.80	0.90	0.90	0.94	0.94
D2	32	0.83	0.82	0.80	0.83	0.83	-	-
D2	64	0.84	0.83	0.81	0.83	0.84	-	-
D2	128	0.84	0.83	0.82	0.84	0.85	0.86	0.85

The analysis of performance results reveals a clear trade-off between computational cost and accuracy, depending on the dataset characteristics. On Musae-Facebook, the GNN baselines (GCN/GAT) achieved the highest performance (94.9% accuracy). This is expected, as this naive $\mathcal{O}(N \times F)$ approach was still computationally feasible for this dataset and could effectively combine input features with the graph structure in a supervised training setting. However, our lightweight approach (KNN with 128-dimensional embeddings) reached a highly competitive peak of 90.1%, demonstrating that feature compression alone already captures most of the predictive information. In the case of Musae-Github, our approach (VGAE@1-epoch + XGBoost with 128-dimensional embeddings) achieved an almost identical performance of 85% compared to the baseline.

To validate our accuracy results and ensure they were not skewed by the class imbalance present in the datasets (as discussed in Section 4.1), we also evaluated the Weighted F1-Score. This metric is crucial because it penalizes models that fail to correctly classify minority classes.

Table 3 presents the the best weighted F1-score results for the Musae Facebook and Musae GitHub datasets, enabling a comprehensive evaluation of how model performance scales with embedding dimensionality while also contrasting embedding-based classifiers against structure-aware GNN architectures.

As shown in Table 3, the Weighted F1-Score results strongly corroborate the trends observed in Table 2 (Accuracy). We note that the F1-Score values are very close to the accuracy values for all top-performing models (e.g., GCN and our approach with XGBoost).

Table 3. Classification Performance Comparison (Weighted F1-Score)

Data	E	MLP	KNN	LR	RF	XGB	GCN	GAT
D1	32	0.77	0.88	0.68	0.85	0.84	-	-
D1	64	0.83	0.90	0.73	0.89	0.87	-	-
D1	128	0.85	0.90	0.80	0.90	0.90	0.94	0.94
D2	32	0.82	0.81	0.80	0.81	0.80	-	-
D2	64	0.83	0.82	0.81(1)	0.82	0.83	-	-
D2	128	0.83	0.83	0.82	0.83	0.84	0.86	0.85

This confirms that the models are achieving robust and balanced performance across all classes, rather than merely predicting the majority class. Since the insights from both metrics are consistent, our main discussion in Section 6 (Conclusion) will refer to these performance results interchangeably, knowing they are validated by both accuracy and F1-Score.

5.2 Computational Performance Results

To quantify the computational efficiency of our lightweight framework, we compared the total Training Time (TT) required by each model. This metric is fundamental to our proposal, which aims to reduce the ”extensive training time” of traditional GNNs. We recorded the CPU processing time for conventional classifiers (MLP, KNN, LR, RF, XGB) operating on precomputed embeddings, as well as the end-to-end training time of the GNN baselines (GCN and GAT). Table [4] presents the training time results (in seconds) for the Musae Facebook and Musae GitHub datasets, enabling a direct analysis of the computational overhead of our decoupled approach compared to GNN architectures that use the graph structure.

Table 4. Training Time Comparison (seconds)

Data	E	MLP	KNN	LR	RF	XGB	GCN	GAT
D1	32	16.23	0.005	78.9	8.15	10.6	-	-
D1	64	16.36	0.004	104.4	12.7	18.5	-	-
D1	128	18.22	0.005	280.4	17.5	33.8	371.8	3566.5
D2	32	21.1	0.004	20.8	14.6	6.3	-	-
D2	64	24.1	0.004	18	24.1	11.88	-	-
D2	128	27.1	0.004	31.7	33.3	21.4	559.7	5514.7

Table 5. Time VGAE (seconds)

Data	E	1 epoch	200 epochs
D1	32	3.1	630.7
D1	64	4.6	875.5
D1	128	6.3	1371.8
D2	32	5.3	1161.6
D2	64	8.6	1560.4
D2	128	11.1	2360.5

When evaluating the feasibility of a graph learning pipeline, runtime is as critical a factor as memory consumption. Our time analysis is divided into two distinct phases, mirroring the architecture of our decoupled approach: (1) the training/embedding generation time and (2) the final classification time.

Time Cost in the Classification Stage. In this phase, the computational savings of our approach become even clearer. Since the classifier receives fixed-size embeddings instead of operating directly on the graph, its complexity depends only

on the number of samples and the embedding dimension, not on the graph topology or the neighborhood aggregation steps that dominate GNN runtimes. As a result, traditional ML classifiers (e.g., MLP, KNN, LR, RF, XGB) achieve competitive performance with orders of magnitude lower training time, as shown in 4. This confirms that, once embeddings are generated, the classification stage of our decoupled pipeline is lightweight, scalable, and substantially faster than end-to-end GNN training.

Approach 1 (Naive GNN): Models like GCN Classifier or GAT Classifier require full supervised training. This involves running dozens or hundreds of epochs (e.g., 200 epochs), where each epoch executes complex message passing operations (neighborhood aggregation) over the entire set of edges E. The time cost is, therefore, high and dependent on the complexity of the GNN model and the number of epochs.

Approach 2 (Our Proposal): Our classification stage feeds the N x D embeddings (which already contain the structural information) into traditional machine learning classifiers. Models like Logistic Regression or k-Nearest Neighbors classifier are computationally much lighter. They operate on a static matrix, do not perform message passing, and in many cases, their training is almost instantaneous (e.g., a single optimization pass or simply data storage).

The classification stage in our approach is orders of magnitude faster than training a supervised GNN end-to-end, as it eliminates the need for neighborhood aggregation at classification time.

A direct comparison of training time between the approaches (Tables 4 and 5) may initially suggest that our framework is disadvantaged, since a VGAE trained for 200 epochs exhibits runtime similar to supervised GNNs such as GCN and GAT. However, this comparison overlooks the fundamental distinction in training objectives. While GNN classifiers must optimize node labels through multiple supervised passes, the VGAE aims only to capture the graph's structural patterns in an unsupervised fashion, a considerably simpler learning task.

Our experiments demonstrate that VGAE training converges rapidly, with embeddings produced after a single epoch already yielding competitive downstream classification performance. Therefore, the practical cost of our pipeline is limited to one epoch of lightweight embedding generation plus a near-instantaneous classifier training step, whereas naive GNNs require hundreds of computationally intensive epochs.

Therefore, when considering the number of epochs realistically required for each task, our decoupled approach proves to be drastically superior not only in memory efficiency but also in total execution time.

Although runtime assessment is important, our results indicate that memory consumption is the dominant bottleneck in large-scale graph learning. In GNN-based architectures, high-cardinality sparse features must be converted into dense multi-hot matrices, increasing the memory footprint to $\mathcal{O}(N \times F)$. This representation is often prohibitive, as feasibility becomes constrained by available RAM rather than computational throughput.

Our memory evaluation therefore examines two critical stages: embedding training and downstream classification. The naive GNN requires allocating dense tensors and storing intermediate activations during message passing, leading to peak usage that frequently exceeds hardware limits. In contrast, our approach relies on sparse data for VGAE training and compact $\mathcal{O}(N \times D)$ embeddings for classification, keeping RAM requirements within practical bounds across all datasets tested.

- Approach 1 (Naive - GNN): Requires the instantiation of a dense multi-hot matrix $\mathcal{O}(N \times F)$, where F is the total number of features in the vocabulary.
- Approach 2 (Efficient - VGAE): Operates on the original sparse data, converting it into tensors of indices and weights for an nn.EmbeddingBag layer $\mathcal{O}(E_{sparse})$

Table 6. Peak Memory Comparison (GB)

Data	E	MLP	KNN	LR	RF	XGB	GCN	GAT
D1	32	0.82	0.66	0.65	0.72	0.84	-	-
D1	64	0.92	0.9	0.83	0.84	0.93	-	-
D1	128	0.97	0.93	0.93	0.94	1	1.46	7.42
D2	32	1	0.93	0.93	0.93	1	-	-
D2	64	1.1	1	1	1	1.1	-	-
D2	128	1.21	1.12	1.15	1.1	1.20	2	11

Table 7. Peak Memory VGAE (GB)

Dataset	E	1	200
D1	32	1.32	1.57
D1	64	1.21	1.63
D1	128	1.48	2
D2	32	1.89	2.16
D2	64	1.84	2.24
D2	128	2.17	3.09

Table 8. Data Load Memory Comparison (MB)

Data	VGAE	E_32	E_64	E_128	GCN/GAT
D1	8.9	2.9	5.6	11.1	409.4
D2	17	4.89	9.4	18.7	585

The memory results (Tables 6-8) highlight the fundamental scalability limitation of naive GNN pipelines. In these models, the memory footprint is not only associated with network parameters but is largely dominated by the storage of dense multi-hot feature representations and the intermediate activations required during message passing. In practice, the peak RAM usage results from the combination of three factors: The base cost of imported libraries, the memory required to load and store data, and the operational memory needed for gradients and temporary tensors during training.

For datasets with large feature vocabularies, the data component alone, which scales with the number of nodes multiplied by the feature dimension, becomes prohibitive. For example, in D2, the GAT model exceeds 11 GB of RAM consumption, frequently leading to out-of-memory failures.

In contrast, our proposed pipeline avoids dense multi-hot encodings. During the embedding stage, VGAE operates directly on sparse structures with cost $\mathcal{O}(E_{\text{sparse}})$. After training, each node is represented by a compact embedding with cost $\mathcal{O}(N \times D)$, where $D \ll F$. As lightweight classifiers require no message passing, the operational cost remains negligible. This design keeps the peak memory usage near 1 GB even on the largest datasets evaluated, dominated mainly by standard library overhead rather than model-specific requirements.

Overall, the results confirm that the memory bottleneck in naive GNNs stems from the dependency on $\mathcal{O}(N \times F)$ data representations. By eliminating this constraint in both training and inference, our decoupled approach remains feasible and efficient under restrictive hardware budgets.

6 Conclusion

In this work, we propose a decoupled learning framework for node classification in large-scale graphs. By structurally separating the representation learning phase from the downstream classification task, our approach effectively yields a lightweight solution designed to address the memory complexity and training time bottlenecks of traditional end-to-end GNNs on large scale datasets.

Our analysis identifies the reliance on raw multi-hot feature representations, with complexity $\mathcal{O}(N \times F)$, as a critical scalability barrier. In contrast, transitioning to compact $\mathcal{O}(N \times D)$ latent embeddings efficiently encodes the interplay between node attributes and local structure. This representation not only reduces dimensionality but also ensures that memory remains manageable as the feature vocabulary expands, making our solution effectively scalable to large and feature-rich graphs. Finally, our investigation aimed to assess the viability of replacing deep GNNs with embedding-based pipelines, particularly for feature-rich datasets like Musae. As indicated in Table 2, even with a single training epoch, the proposed method retained 98% of the predictive capacity of a full GCN (85.1% vs. 86.7%). Regarding computational feasibility, the combined time for embedding generation (Table 5) and classification (Table 4) was approximately 16 times lower than the GCN baseline (34s vs. 560s in Table 4). These results support the premise that the decoupled $\mathcal{O}(N \times D)$ architecture offers a practical and scalable trade-off for large-scale graph analysis.

As our results indicated that attribute information plays a major role in predictive performance, a crucial next step is to evaluate the framework on datasets where structural relationships are the primary source of information, allowing a deeper assessment of the effectiveness of the VGAE's self-supervised learning. Additionally, future investigations should consider scenarios in which new nodes arrive over time and embeddings must be generated without retraining the entire model, thereby assessing its generalization capacity in dynamic graph settings.

In addition, expanding the experiments to larger datasets and more optimized models, including a more extensive hyperparameter search for the models, and other techniques, would help refine the results while preserving relevant information and further reducing computational cost.

Acknowledgements

The authors would like to thank the CNPq (N^o 305946/2025-0 and N^o 405940/2022-0) and CAPES (N^o 88887.954253/2024-00 and N^o 88887.972043/2024-00) of Brazil for the financial support.

Funding

The authors would like to thank the CNPq (N^o 305946/2025-0 and N^o 405940/2022-0) and CAPES (N^o 88887.954253/2024-00 and N^o 88887.972043/2024-00) of Brazil for the financial support.

Authors' Contributions

All authors contributed to the development of the proposal and the writing of this manuscript.

Competing interests

The authors declare no conflicts of interest.

Availability of data and materials

The code developed, as well as the data used in the experiments, are available in the project repository (`https://github.com/LarcesUece/gnns_vs_vgae_classifier`) and with the necessary instructions for reproducibility.

References

Brochier, R., Guille, A., and Velcin, J. (2019). Global vectors for node representations. In *The World Wide Web Conference*, pages 2587–2593. DOI: 10.1145/3308558.3313595.

Cai, H., Zheng, V. W., and Chang, K. C.-C. (2018). A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE transactions on knowledge and data engineering*, 30(9):1616–1637. DOI: 10.1109/tkde.2018.2807452.

Chen, Z., Tan, H., Wang, T., Shen, T., Lu, T., Peng, Q., Cheng, C., and Qi, Y. (2023). Graph propagation transformer for graph representation learning. *arXiv preprint arXiv:2305.11424*. DOI: 10.24963/ijcai.2023/396.

Costa, W. L., Silveira, M. M., de Araujo, T., and Gomes, R. L. (2020). Improving ddos detection in iot networks through analysis of network traffic characteristics. In *2020 IEEE Latin-American Conference on Communications (LATINCOM)*, pages 1–6. IEEE. DOI: 10.1109/latincom50620.2020.9282265.

Dalvi, A. and Honavar, V. (2025). Hyperdimensional representation learning for node classification and link prediction. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, pages 88–97. DOI: 10.1145/3701551.3703492.

Epasto, A. and Perozzi, B. (2019). Is a single embedding enough? learning node representations that capture multiple social contexts. In *The world wide web conference*, pages 394–404. DOI: 10.1145/3308558.3313660.

Fortunato, S. (2010). Community detection in graphs. *Physics reports*, 486(3-5):75–174. DOI: 10.1016/j.physrep.2009.11.002.

Goyal, P., Kamra, N., He, X., and Liu, Y. (2018). Dyngem: Deep embedding method for dynamic graphs. *arXiv preprint arXiv:1805.11273*. DOI: 10.48550/arxiv.1805.11273.

Heidari, F. and Papagelis, M. (2020). Evolving network representation learning based on random walks. *Applied network science*, 5(1):18. DOI: 10.1007/s41109-020-00257-3.

Khan, I., Bokhari, M. U., Afzal, S., and Alam, S. (2025). Distance driven graph neural network for advanced node classification through feature augmentation. *Discover Computing*, 28(1):70. DOI: 10.1007/s10791-025-09569-3.

Khoshraftar, S. and An, A. (2024). A survey on graph representation learning methods. *ACM Transactions on Intelligent Systems and Technology*, 15(1):1–55. DOI: 10.1145/3633518.

Kipf, T. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*. DOI: 10.48550/arxiv.1609.02907.

Lecca, P. and Lecca, M. (2023). Graph embedding and geometric deep learning relevance to network biology and structural chemistry. *Frontiers in Artificial Intelligence*, 6:1256352. DOI: 10.3389/frai.2023.1256352.

Li, S., Zaidi, N. A., Du, M., Zhou, Z., Zhang, H., and Li, G. (2024). Property graph representation learning for node classification. *Knowledge and Information Systems*, 66(1):237–265. DOI: 10.1007/s10115-023-01963-x.

Lin, S., Hong, J., Lang, B., and Huang, L. (2023). Dag: Dual attention graph representation learning for node classification. *Mathematics*, 11(17):3691. DOI: 10.3390/math11173691.

Luo, Y., Liu, Q., Shi, L., and Wu, X.-M. (2024). Structure-aware semantic node identifiers for learning on graphs. *arXiv e-prints*, pages arXiv–2405. Available at:`https://arxiv.org/html/2405.16435v1`.

Mahdavi, S., Khoshraftar, S., and An, A. (2019). Dynamic joint variational graph autoencoders. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 385–401. Springer. DOI: 10.1007/978-3-030-43823-4$_3$2.

Makarov, I., Kiselev, D., Nikitinsky, N., and Subelj, L. (2021). Survey on graph embeddings and their applications to machine learning problems on graphs. *PeerJ Computer Science*, 7:e357. DOI: 10.7717/peerj-cs.357.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26. DOI: 10.48550/arxiv.1310.4546.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., *et al*. (2011). Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830. DOI: 10.48550/arxiv.1201.0490.

Pimenta, I., Silva, D., Moura, E., Silveira, M., and Gomes, R. L. (2024). Impact of data anonymization in machine learning models. In *Proceedings of the 13th Latin-American Symposium on Dependable and Secure Computing*, pages 188–191. DOI: 10.1145/3697090.3699865.

Pimenta, I. A., Lee, M. H., Bittencourt, L. F., and Gomes, R. L. (2025). Adaptive privacy based on mutual information for machine learning in edge-cloud environments. *IEEE Networking Letters*. DOI: 10.1109/lnet.2025.3607555.

Rozemberczki, B., Davies, R., Sarkar, R., and Sutton, C. (2019). Gemsec: Graph embedding with self clustering. In *Proceedings of the 2019 IEEE/ACM international conference on advances in social networks analysis and mining*, pages 65–72. DOI: 10.48550/arxiv.1802.03997.

Silveira, M. M., Silva, D. S., Rodriguez, S. J., and Gomes, R. L. (2022). Searchable symmetric encryption for private data protection in cloud environments. In *Proceedings of the 11th Latin-American Symposium on Dependable Computing*, pages 95–98. DOI: 10.1145/3569902.3570171.

Wang, D., Cui, P., and Zhu, W. (2016). Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234. DOI: 10.1145/2939672.2939753.

Wang, J., Peng, J., Huang, F., Liao, S., Zhan, P., and Yi, P. (2025). Information enhancement graph representation learning. *Pattern Recognition Letters*. DOI: 10.1016/j.patrec.2025.04.006.

Wang, X., Cui, P., Wang, J., Pei, J., Zhu, W., and Yang, S. (2017). Community preserving network embedding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31. DOI: 10.1609/aaai.v31i1.10488.

Wei, X., Xu, L., Cao, B., and Yu, P. S. (2017). Cross view link prediction by learning noise-resilient representation consensus. In *Proceedings of the 26th international conference on World Wide Web*, pages 1611–1619. DOI: 10.1145/3038912.3052575.

Xiao, W., Zhao, H., Zheng, V. W., and Song, Y. (2020). Vertex-reinforced random walk for network embedding. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pages 595–603. SIAM. DOI: 10.1137/1.9781611976236.67.

Zhao, M., Huang, X., Lyu, Z., Wang, Y., Cui, L., and Bai, L. (2024). Knowledge probing for graph representation learning. *arXiv preprint arXiv:2408.03877*. DOI: 10.48550/arxiv.2408.03877.

Zhu, H. and Koniusz, P. (2021). Refine: Random range finder for network embedding. In *Proceedings of the 30th ACM international conference on information & knowledge management*, pages 3682–3686. DOI: 10.48550/arXiv.2108.10703.