RESEARCH PAPER DOI: 10.22080/cste.2026.31137.1115

Contributions of Science and Technology for Engineering

Journal Home page: https://cste.journals.umz.ac.ir/

An Efficient Hybrid GCN-DDPG Model for Node Classification in Large-Scale Graphs

Fatemeh Fathinezhad [1]*

[1] *Department of Computer Engineering and Information Technology, Razi University, Kermanshah Province, Iran, ORCID: 0000-0002-2474-9644.*

**Corresponding Author: f.fathinezhad@razi.ac.ir*

Article Info

Received 28 January 2026

Accepted 05 November 2026

Available online 08 November 2026

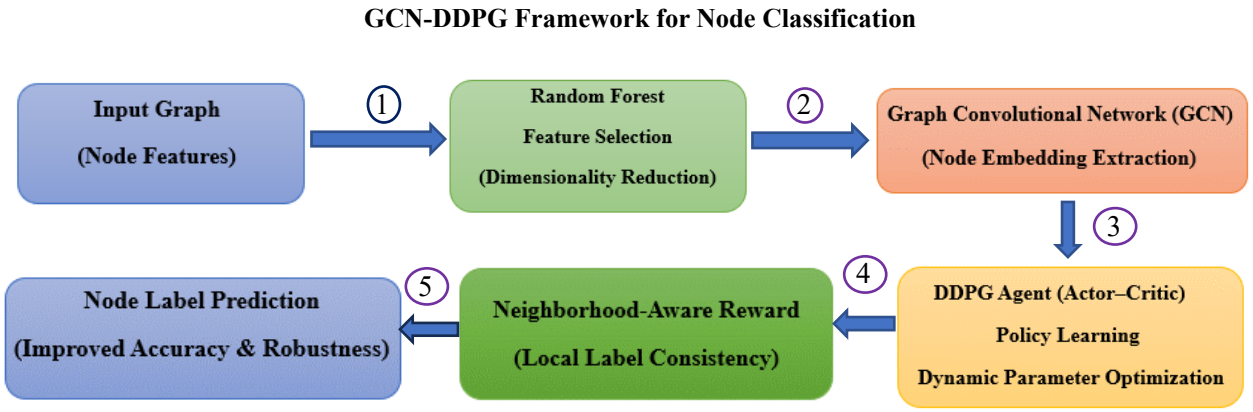
Keywords:

Graph Convolutional Networks;
Deep Reinforcement Learning;
Deep Deterministic Policy Gradient;
Node Classification;
Neighborhood Consistency.

Abstract:

Graph-structured data present significant challenges for machine learning due to their non-Euclidean nature, complex topological dependencies, and heterogeneous node features. This paper proposes a hybrid framework that integrates graph convolutional networks (GCNs) with the deep deterministic policy gradient (DDPG) algorithm to improve node classification in large-scale graphs. Unlike conventional GCNs with fixed feature aggregation, the proposed model employs a reinforcement learning agent based on continuous-action representations that operates on GCN-generated node embeddings and learns a policy to map these embeddings to class preference vectors using a reward signal. To enhance efficiency and stability, a random forest-based feature selection mechanism is incorporated, and a neighborhood-aware reward function is designed to enforce label consistency among connected nodes. Experiments conducted on five benchmark datasets ,Cora, Citeseer, PubMed, Twitch-Gamers, and FacebookPagePage ,demonstrate that the proposed GCN-DDPG model consistently outperforms baseline methods, including GCN, GAT, and other GCN–reinforcement learning variants. The proposed framework achieves relative accuracy improvements of +11.91% on PubMed, +10.84% on Citeseer, +9.06% on Cora, and +5.57% on Twitch compared to the baseline GCN model, while exhibiting low variance across repeated runs, indicating strong robustness. Overall, the results confirm that continuous-control reinforcement learning, when properly aligned with graph neural architectures and reward design, provides an effective and scalable solution for high-fidelity node classification in complex graph-structured data.

Graphical Abstract



© 2026 University of Mazandaran

Copyright © 2026 by the authors. Licensee CSTE, Babolsar, Mazandaran. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC-BY) license (https://creativecommons.org/licenses/by/4.0/deed.en)

ISSN 3060-6578

Please cite this paper as: F. Fathinezhad, "An Efficient Hybrid GCN-DDPG Model for Node Classification in Large-Scale Graphs", Contrib. Sci. & Tech Eng, vol. 3, no. 3, pp. 61-74, 2026, doi:10.22080/cste.2026.31137.1115.

1. Introduction

Graph-structured data has become increasingly prevalent across a wide range of domains, including bioinformatics, biochemistry, image processing, recommender systems, and social network analysis [1, 2]. To address the challenges posed by this data, numerous learning models have been developed using various preprocessing techniques. Yet, many of these methods lack the adaptability needed for diverse datasets and tasks. Additionally, the high dimensionality and non-Euclidean nature of graph data present ongoing challenges in machine learning [3].

Conventional neural networks (CNNs) are designed for Euclidean data, limiting their ability to handle irregular graph structures. Graph Neural Networks (GNNs) address this limitation by directly operating on graph-structured data and capturing both local and global dependencies through message passing. As a result, GNNs have become powerful tools for tasks such as node classification, link prediction, graph-level regression, and clustering [4].

The primary goal of graph representation learning is to generate a set of meaningful features that reflect both the structural topology and attribute information of a graph. This involves learning an optimal mapping that embeds each node into a low-dimensional vector space while preserving the geometric relationships of the original graph. While traditional deep learning models have demonstrated great success in processing Euclidean data such as images, speech, and video, recent advances have sparked increasing interest in extending deep learning to non-Euclidean domains through geometric deep learning approaches [5]. However, applying general neural network methods to graph-structured data introduces significant challenges. These include variable node ordering and graph size, irregular topologies, and dynamic neighborhood structures, which hinder the direct use of standard operations such as convolution. Graph Neural Networks are specifically designed to address these issues, typically comprising two main phases: a message-passing phase, where each node aggregates information from its neighbors to capture local structural patterns, and a readout phase, which summarizes the node-level representations into a comprehensive graph-level feature vector [6].

In large and complex graphs, scalability becomes a major challenge. Such graphs often require significant computational resources and memory due to their high dimensionality and complex structure. Among the advanced methods for processing graph data, Graph Convolutional Networks have emerged as effective tools. These models can efficiently extract local node features and simulate the hidden structures within graphs [7].

GCNs are particularly recognized as powerful approaches for node classification tasks in complex graph structures. However, many real-world graphs require modeling long-range dependencies or indirect relationships among nodes, which traditional models struggle to capture. In these cases, the integration of reinforcement learning offers a promising solution by enabling sequential decision-making and capturing deeper node interactions across the graph [8].

While GCNs are suitable for many graph structures and problems, they may not perform optimally on more complex graphs or those requiring specialized processing. Therefore, combining GCNs with different learning methods can significantly improve accuracy. In general, GCNs, due to their ability to capture complex relationships between data points, can provide better inputs for reinforcement learning models. The use of deep neural networks to solve sequential decision-making problems within the framework of reinforcement learning has led to the emergence of deep reinforcement learning in high-dimensional problems [9].

Reinforcement learning has been widely adopted to enhance decision-making in complex environments, and actor–critic methods such as the Asynchronous Advantage Actor–Critic (A3C) have demonstrated strong performance in large-scale, dynamic tasks by enabling parallel policy updates across multiple environments [10]. A3C's ability to model evolving patterns makes it a useful approach for graph-related problems, where relationships between nodes may change or exhibit non-stationary behavior. To further improve stability in sequence-dependent settings, A3C is often combined with Long Short-Term Memory (LSTM) networks, which effectively capture long-range temporal dependencies and mitigate information loss over time [11, 12].

We propose a hybrid framework that combines GCNs with DDPG to perform node classification on large-scale, complex graph structures effectively. Initially, node features are selected using the random forest algorithm [13], enabling the model to focus on the most informative features and thereby enhancing GCN performance. The GCN then processes the refined graph data to extract meaningful node representations. These representations are subsequently passed to the DDPG agent, which learns optimal decision policies for node label prediction. This integrated GCN-DDPG framework proves particularly effective for large-scale, complex graphs characterized by nonlinear, long-range dependencies among nodes. The reward function is designed to incentivize accurate predictions for both target nodes and their neighbors, thereby facilitating simultaneous learning of node representations and precise modeling of local graph structures. The main strategic insights underlying this research framework, which collectively inform the proposed solution, are as follows:

1. Integrating GCN feature extraction with DDPG-based decision learning: This study combines the representation learning ability of graph convolutional networks with the policy optimization capability of the deep deterministic policy gradient algorithm to enhance node classification in large and complex graphs. In the proposed framework, the GCN extracts structural and semantic features from the graph data, which are then used by the DDPG agent to support decision-making for node label assignment. The actor network in the DDPG structure produces a continuous output vector (via a tanh activation), which is then mapped to a discrete class label by applying argmax over the output values. This integration enables the model to capture both local structural dependencies and high-level decision patterns, improving classification accuracy compared to conventional GCN and reinforcement learning approaches.

2. Feature dimensionality reduction via random forest optimization: To address the challenges of high-dimensional graph features, a random forest-based feature importance analysis is incorporated before GCN processing. This mechanism filters

out redundant attributes and retains only the most informative ones, reducing computational cost and accelerating convergence while preserving classification performance on heterogeneous graphs.

3. Locally-consistent reward function for structural preservation: A novel neighborhood-aware reward function is designed to encourage label consistency among connected nodes. Beyond rewarding correct predictions for individual nodes, this formulation promotes agreement within local neighborhoods, explicitly embedding graph topology into the reinforcement learning signal. As a result, the model achieves stronger local structural coherence and higher classification accuracy.

4. Dynamic reinforcement-based parameter optimization: The DDPG component is employed to progressively update the parameters of the graph model based on the feedback received from the environment through the proposed reward function. This integration allows the model to continuously adjust its policy and value estimates during training, leading to more stable convergence and improved decision quality. By combining GCN's representational power with DDPG's continuous policy learning, the framework achieves adaptive optimization of model parameters through reinforcement signals, thereby enhancing the model's capability to handle complex and non-stationary graph structures.

5. Comprehensive evaluation on large-scale real-world graphs: The proposed GCN–DDPG framework is thoroughly evaluated on several large and heterogeneous real-world graph datasets, including Citeseer, PubMed, Cora, Twitch-Gamers, and Facebook-PagePage. These datasets cover a wide range of node and edge densities, feature dimensions, and class distributions. Through this comprehensive evaluation, the study assesses the generalizability, scalability, and robustness of the proposed model under diverse structural and statistical graph conditions, confirming its capability to perform effectively in non-uniform and complex environments.

The structure of the paper is as follows: Section 2 reviews the related literature and prior research. Section 3 presents the technical background on graph convolutional networks and reinforcement learning algorithms, with a focus on DDPG and A3C. Section 4 details the proposed methodology, while Section 5 discusses the experimental results. Section 6 examines the impact of the number of layers and the hidden feature dimension. Section 7 analyzes the time complexity and discusses the model's limitations. Finally, Section 8 concludes the paper by summarizing the key findings.

2. Related Work

In deep reinforcement learning, unstructured input data is directly provided to the network from the state space. These inputs, which may include high-dimensional data such as game screen pixels, camera images, or raw sensor streams from a robot, are often large and complex. The network's output determines the value of each action, enabling the agent to decide on the optimal actions to take in the environment to maximize expected rewards. Given that reinforcement learning methods face challenges in high-dimensional spaces, deep neural networks can automatically extract low-dimensional representations and features from such data. Several deep reinforcement learning models have been developed to address these challenges, including Deep Q-Networks (DQN), Double Deep Q-Networks (DDQN), and Dueling DQN [14, 15], among others.

As described by Leng et al. [16], the proposed production control system is designed to be resilient against disruptions such as failures, delays, and order changes, making it suitable for large-scale operations in advanced industrial contexts. This system leverages a combination of deep reinforcement learning and graph convolutional networks. In this framework, production elements (such as machines, operations, parts, etc.) are modeled as nodes, and the relationships between them (including temporal and functional dependencies) are represented as edges. GCN is employed to extract hidden features and capture the interconnections between production components, helping the model understand the production structure. To enable resilient and adaptive control, the GCN outputs are passed to a deep reinforcement learning agent such as DQN or DDPG that optimizes decision-making in dynamic and uncertain production environments. The proposed method has shown superior performance over conventional techniques and certain machine learning models, especially in managing uncertainty and handling various perturbations. In Rokhforoz et al.'s study [17], a semi-distributed learning algorithm is introduced that combines Deep Reinforcement Learning (DRL) with Graph Convolutional Networks (GCNs) to determine optimal bidding strategies for generation units in electricity markets under uncertainty. In this approach, a GCN is used to extract latent features and capture the topological structure of the power grid, in which generation units and their interdependencies are modeled as graph nodes and edges. The features extracted by the GCN are then passed to a DRL agent, which learns optimal bidding strategies through interaction with the environment and continuous feedback. This integrated framework enables generation units to adapt their strategies dynamically under uncertainty, leading to increased profitability and improved market performance compared to traditional and machine-learning-based approaches.

Gao et al. [18] proposed a graph convolution-based deep reinforcement learning framework for decision-making in uncertain and interactive traffic scenarios. In their approach, Graph Convolutional Networks (GCNs) are employed to extract meaningful representations from graph-structured traffic environments. The extracted features are subsequently incorporated into Deep Q-Network (DQN)-based learning to determine appropriate decision-making policies. The proposed framework enables agents to capture interactions and dependencies among surrounding vehicles. Its effectiveness is evaluated through experiments involving autonomous driving and interactive traffic scenarios. The results demonstrate the potential of combining graph representation learning with deep reinforcement learning to improve decision-making in complex and dynamic environments.

In Weng et al.'s study [19], a hybrid framework is proposed for optimizing bidding strategies in electricity markets. This framework leverages GCNs to extract structural information from the power grid and uses the DDPG algorithm to learn the optimal bidding policy. The model is capable of making better decisions in dynamic, real-time conditions by accounting for complex graph dependencies among market factors. Experimental results demonstrate that the proposed method outperforms baseline approaches in terms of accuracy, stability, and profitability.

In the study by Wu et al. [20], an intelligent framework is proposed to simultaneously optimize drone paths and task offloading in aerial edge computing environments. The problem environment is modeled as a graph, where Graph Convolutional Networks (GCNs) are employed to extract spatial and topological features of the system. To learn the decision-making policy, a DDPG-based deep reinforcement learning algorithm is used, with a continuous actor-critic architecture designed for precise decisions in the continuous action space. Simulation results show that this framework reduces processing delays and improves the overall system performance compared to traditional methods. The integration of GCN with DDPG enables the model to effectively learn and incorporate structural dependencies in the environment, thereby enhancing decision-making.

The main innovation of Nurbek [21] lies in integrating Graph Neural Networks (GNNs) with RLs to address mobility problems. Unlike traditional methods, which typically rely on matrix or sequential features for learning, this study demonstrates that GNNs can effectively process complex graph structures and model the relationships between nodes and edges. These graph-based features are particularly beneficial in dynamic and complex environments, contributing to improved learning of optimal policies. The results show that GNN architectures, specifically optimized for graph processing, outperform traditional models in mobility tasks. However, while this highlight the success of GNNs in certain mobility tasks, it also suggests that these methods may not achieve optimal performance for tasks with different characteristics. In other words, the generalizability of these methods to other environments and tasks may be limited.

In multi-agent reinforcement learning, a critical challenge is enabling effective communication and cooperation between agents. GNNs have been widely adopted for this purpose, as they facilitate agent interaction through a shared graph structure. However, a major limitation of many existing approaches is the limited extraction of meaningful information from neighboring agents, often due to overlooking topological relationships within the graph. To address this challenge, the method proposed in Ding et al.'s study [22] seeks to enhance the process by maximizing the extraction of relevant information from neighboring agents in the graph structure. The approach focuses on improving the correlation between neighboring agents' input features and their learned hidden representations, thereby enabling more robust feature embeddings. By doing so, the method not only boosts feature extraction efficiency but also optimizes agents' collaborative efforts in performing complex tasks. The proposed method demonstrates notable enhancements in multi-agent coordination, particularly in settings where the underlying interaction topology critically influences task outcomes.

In Guo et al.'s study [23], the authors propose an advanced routing optimization approach that leverages the strengths of GCNs and DDPG reinforcement learning to improve routing performance in Software-Defined Networks (SDNs). Their method uses GCNs to capture the network's dynamic topology and structural features, which inform the DDPG agent's routing decision-making. Experimental evaluations demonstrate that this hybrid approach significantly reduces end-to-end delay and packet loss compared to traditional algorithms such as OSPF and other reinforcement learning-based methods. This work illustrates the potential of integrating graph-based feature extraction with deep reinforcement learning to address complex network routing challenges. Although GCN-DDPG frameworks have shown promising results in addressing dynamic optimization problems in MEC and network routing, they often suffer from limited scalability, reliance on synthetic data, and insufficient modeling of real-world complexities. The lack of robustness in diverse network conditions and the absence of practical validation limit their applicability in real deployments. To address the joint partial offloading and resource allocation (JPORA) problem in energy harvesting (EH)-enabled mobile edge computing (MEC) systems, a centralized GCN-DDPG agent is proposed in Chen and Wu study [24], which integrates Graph Convolutional Networks (GCN) with Deep Deterministic Policy Gradient (DDPG). While DDPG enables continuous control, GCN captures structural dependencies among mobile devices (MDs). The agent optimizes the offloading ratio, local computation, and uplink power allocation. Experimental results demonstrate that GCN-DDPG significantly outperforms existing DRL-based methods in terms of energy efficiency and system performance. The study by Peng et al. [25] introduces GraphRARE, a reinforcement learning-enhanced graph neural network that uses relative entropy as a reward to guide adaptive representation learning. This method effectively balances exploration and exploitation in the learning process. Experimental evaluations show that GraphRARE outperforms traditional GNN models on several benchmark datasets.

Based on Table 1, most existing GCN–RL approaches, particularly those using DDPG, have primarily focused on optimization and control tasks such as process control, energy trading, and routing. As summarized in Table 1, these methods generally rely on predefined input features and lack any explicit feature selection mechanism, making them dependent on manually engineered representations. Consequently, they may suffer from feature redundancy and limited scalability when applied to high-dimensional or complex graph-structured data.

Table 1. Comparison of GCN-RL frameworks by task, reward, features, and domain

Reference	RL Algorithm	Task Type	Reward Function	Feature Selection	Domain
Leng et al. [16]	DDPG	Process Control	Resilient production	None	Manufacturing
Rokhforoz et al. [17]	DRL	Market Bidding	Profit maximization	None	Smart Grid
Weng et al. [19]	DDPG	Energy Bidding	Economic efficiency	None	Electricity Market
Wu et al. [20]	DDPG	Trajectory Planning	Latency minimization	None	Aerial Edge Computing
Guo et al. [23]	DDPG	Routing Optimization	Network latency	None	SDN Routing
Ours	DDPG	Node Classification	Local consistency	Random Forest	Complex Graphs

In contrast, the proposed framework addresses node classification in complex graphs by integrating a random forest–based feature-selection stage before reinforcement learning. This allows the model to automatically identify the most informative features, reduce redundancy, and improve the quality of the input representation. Moreover, the proposed method unifies feature selection, graph representation learning via GCN, and policy optimization using DDPG within a single framework. The graph convolutional network captures structural dependencies among nodes, while the DDPG agent learns an adaptive decision policy guided by a reward function based on local consistency. This integrated design effectively leverages both structural and attribute information to enhance learning efficiency, robustness, and overall performance.

3. Technical Background

In this section, we review the foundational concepts of Graph Convolutional Networks (GCNs), Deep Deterministic Policy Gradient (DDPG), and the Asynchronous Advantage Actor-Critic (A3C) deep reinforcement learning framework. These concepts are essential to understanding the proposed methodology and its application in tasks involving graph-structured data and decision-making in dynamic environments.

3.1. Graph Convolutional Networks (GCNs)

GCNs are designed to handle graph-structured data by learning node embeddings through convolutional operations over graph topology. These networks have proven effective for various tasks such as node classification, link prediction, and graph representation learning.

A graph is fundamentally composed of nodes (vertices) and edges (connections), and can be represented as $G = (\mathrm{V}, E)$, where V denotes the set of nodes and E represents the set of edges. A feature matrix further describes the structure of a graph X and an adjacency matrix A. Let us assume that the graph has N nodes, and each node i has f features, which are denoted as x_i for $i = 1, \dots, N$. In this case, the feature matrix X has dimensions $N \times f$, where each row i of X corresponds to the feature vector of node ii. The adjacency matrix A is an $N \times N$ matrix representing the connectivity of the graph. If node i is connected to node j, then the entry in row $i$and column j of A is 1; otherwise, it is 0, indicating the absence of an edge between the two nodes [25].

The input to a single layer of a GCN is the pair $\{X, A\}$, which consists of the feature matrix X and the adjacency matrix A. For a GCN with L layers, the output of layer l, where $l = 1, \dots, L$, can be defined as a nonlinear function of the input:

For each layer l in a Graph Convolutional Network GCN, the feature update relation is defined as:

$$O^{l+1} = g(O^l, A) \tag{1}$$

where $O^0 = X$, meaning the input to the first layer is the feature matrix X. The non-linear function g is defined as:

$$g(O^l, A) = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}W^l) \tag{2}$$

In this relation, W^l is the weight matrix corresponding to layer l, σ is an activation function like ReLU, $\hat{A} = \mathrm{A} + \mathrm{I}$ is the adjacency matrix with the addition of an identity matrix I, and $\widehat{D}$ is the diagonal degree matrix of the nodes in $\hat{A}$. This relation allows for the propagation of information from neighboring nodes and updates the node features based on the graph structure and local features.

3.2. Deep Deterministic Policy Gradient (DDPG)

DDPG is an actor-critic reinforcement learning algorithm suited for continuous action spaces. It employs deep neural networks for both the actor (policy) and critic (value function) and uses experience replay and target networks to stabilize learning. The DDPG algorithm, by combining a deterministic policy, off-policy learning, experience replay, and target networks, is a powerful and stable reinforcement learning method for continuous action spaces. This method is especially widely applied in control problems and decision optimization.

DDPG is an advanced reinforcement learning method based on policy that is designed for continuous action spaces and utilizes an actor-critic structure. This algorithm operates off-policy and is a deterministic version of the Deterministic Policy Gradient (DPG) algorithm, implemented using deep networks. In DDPG, two separate neural networks are defined.

The actor network models the deterministic policy and generates an action $a = \mu(s \mid \theta_\mu)$ for each state s. The critic network estimates the value function $Q(s, a \mid \theta_v)$ and evaluates the quality of actions. To reduce temporal correlation between samples and improve training stability, the agent's experiences are stored as tuples (s, a, r, s') in an experience replay memory and are sampled randomly for training. For increased stability, DDPG uses target networks for both the actor and critic networks, which are slower, softened copies of the main networks.

$$\theta\mu' \leftarrow \tau\theta_\mu + (1 - \tau)\theta\mu' \tag{3}$$

$$\theta_{v'} \leftarrow \tau\theta_v + (1 - \tau)\theta_{v'} \tag{4}$$

where $\tau \ll 1$ is the soft update rate. The critic network is updated by minimizing the mean squared error (MSE) loss function:

$$L(\theta_v) = E_{D\sim(s,a,r,s')}[(Q(s, a \mid \theta_v) - y)^2] \tag{5}$$

where the target value y is defined as follows:

$$y = r + \gamma Q'\left(s', \mu'\left(s' \mid \theta_{\mu'}\right) \mid \theta_{v'}\right) \tag{6}$$

In reinforcement learning, the replay buffer (denoted by D) stores the agent's experiences during its interaction with the environment. During training, these experiences are randomly sampled to reduce temporal correlation in the data and improve the stability of the learning process. The actor network is trained by maximizing the expected Q-value, with the policy aiming to produce actions the critic evaluates as highly valuable. The objective function for the actor is defined as:

$$J(\theta_\mu) = E_{s\sim D}[\nabla_a\, Q(s, \mu(s|\theta_\mu)] \tag{7}$$

To optimize this objective, the gradient with respect to the actor's parameters is computed using the deterministic policy gradient theorem:

$$\nabla_{\theta_\mu} J(\theta_\mu) \approx E_{s\sim D}\left[\nabla_a Q(s, a \mid \theta_v)\mid_{a=\mu(s)} \nabla_{\theta_\mu}\mu(s \mid \theta_\mu)\right] \tag{8}$$

In this framework, the actor learns to generate actions that maximize the estimated Q-values, thereby improving the quality of the selected actions over time.

4. Proposed Method

We propose an innovative hybrid framework for node classification in complex social graphs that combines the representational power of GCNs with the decision-making capabilities of the DDPG algorithm. This approach addresses key limitations of conventional graph-processing methods and improves classification accuracy. The pseudocode of the proposed method is presented in Algorithm 1. The key steps of the proposed method are outlined below:

Algorithm 1 GCN-DDPG for Node Classification (Contextual Bandit)

Require: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, node features X, adjacency matrix A, labels y, training nodes $\mathcal{V}_{train}$
Ensure: Predicted labels $\hat{y}$

Initialize GCN parameters θ_g
Initialize actor π_θ and critic Q_ϕ
for $run = 1$ **to** N_{runs} **do**
  Set random seed
  Preprocess data (normalization, feature selection)
  for $epoch = 1$ **to** M **do**
    // **Step 1: Context (state) construction**
    $S \leftarrow \text{GCN}(A, X; \theta_\phi)$
    // **Step 2: Action selection**
    $a_{pred}, \hat{Q} \leftarrow \text{DDPG}(S)$ ($\hat{a}_{pred}$: predicted action scores)
    $\hat{y} \leftarrow \arg\max(\hat{a}_{pred})$
    // **Step 3: Reward computation**
    for each node $i \in V_{train}$ **do**
      if $\hat{y}_i = y_i$ **then**
        $r_i \leftarrow 1$
      else
        if $\exists j \in \mathcal{N}(i)$ such that $\hat{y}_j = y_j$ **then**
          $r_i \leftarrow \alpha$
        else
          $r_i \leftarrow 0$
        end if
      end if
    end for
    // **Step 4: Loss computation**
    $L_{GCN} \leftarrow \text{CrossEntropy}(A_{pred}, \mathcal{V}_{train}, y, \mathcal{V}_{train})$
    $L_{DDPG} \leftarrow \text{MSE}(\hat{Q}, r)$
    $L_{total} \leftarrow L_{GCN} + L_{DDPG}$
    Update θ_g, θ, ϕ using ∇L_{total}
  end for
end for
return trained model and $\hat{y}$

4.1. Data Preprocessing and Feature Selection

In the first step, the datasets are loaded. The input features are normalized using the Z-Score standardization method. This technique reduces the impact of varying feature scales, facilitating model learning and minimizing bias during training. After normalization, to reduce feature dimensionality and eliminate irrelevant information, the random forest algorithm is employed for feature selection. Random Forest selects the most important features based on their statistical importance for predicting the final labels and passes them as the final input to the model.

To evaluate the impact of the random forest feature-selection step applied before GCN processing, Table 4 includes both "GCN" and "GCN with feature selection," as well as the proposed GCNDDPG method with and without random forest.

4.2. Graph Feature Extraction Using GCN

To learn node embeddings in the graph, a network architecture consisting of two GCN layers, batch normalization, and a dropout layer to prevent overfitting is employed. This component is responsible for extracting high-level graph features. By leveraging the graph's topological structure and its adjacency matrix, the model incorporates each node's local neighborhood in the feature learning process. As a result, the output of this component consists of lower-dimensional vectors enriched with semantic and structural information, which are directly passed to the reinforcement learning model. The architecture of this network is illustrated in Figure 1.

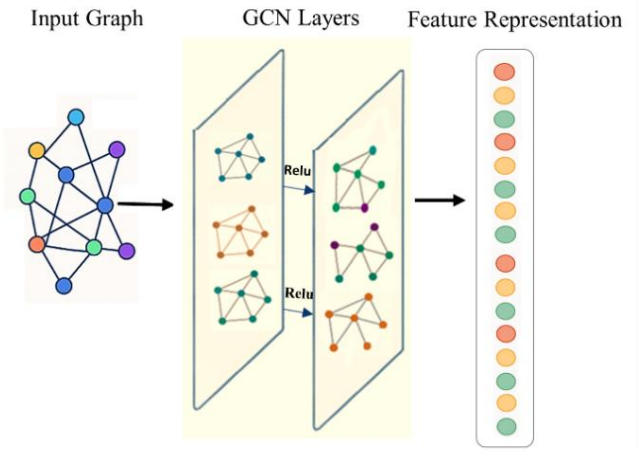


Figure 1. Node embeddings using GCN layers

It is important to note that this formulation corresponds to a contextual bandit setting rather than a full Markov Decision Process (MDP), since the graph is static and no state transitions occur [26, 27]. For each node v_i, the state s_i is defined as the node embedding produced by the GCN encoder:

$$s_i = \text{GCN}(x_i, \mathcal{N}(i)) \tag{9}$$

where x_i denotes the raw node features and $\mathcal{N}(i)$ represents the neighborhood of node i obtained from the adjacency matrix A. The resulting embeddings integrate both structural and feature information through message passing. Consequently, the state corresponds to the learned node representation rather than the raw graph features. Structural dependencies among nodes are captured implicitly through the GCN encoder and the neighborhood-aware reward function, which promotes local label consistency, rather than through explicit state transitions or sequential decision dynamics. Accordingly, node classification decisions are conditionally independent given the learned embeddings, and the actor network processes node representations in parallel.

4.3. Reinforcement Modeling Using DDPG

After extracting graph-based features with the GCN, the resulting node embeddings serve as the context for the reinforcement learning agent. In the proposed formulation, each node embedding represents the state information available to the agent at decision time, while the predicted class label is treated as the corresponding action. This design allows the node classification problem to be framed as a contextual bandit setting, in which the agent learns a policy that maps contextual information to class-selection decisions.

The DDPG module consists of two components, an actor network and a critic network. The actor network $\pi_\theta$takes the node embedding s_i as input and outputs a continuous action vector over the class space:

$$a_i = \pi_\theta(s_i) \tag{10}$$

where $a_i \in \mathbb{R}^C$denotes the class preference scores for Cclasses. Since the final prediction must be discrete, the predicted label is obtained by selecting the class with the maximum score:

$$\hat{y}_i = \arg\max(a_i) \tag{11}$$

The critic network estimates the value of the selected action under the given context. Both networks are implemented using fully connected layers with ReLU activations, while the actor output layer uses a Tanh activation to produce bounded continuous outputs.

This actor–critic structure enables the model to perform label selection and value estimation jointly. During training, the actor is optimized using the classification objective, while the critic is trained to estimate the corresponding value targets. The overall optimization therefore combines supervised classification learning with reinforcement-learning feedback, allowing the model to benefit from both node-level label supervision and context-dependent decision-making. The detailed reward design, including the neighborhood-aware reward that encourages local structural consistency, is presented in the following subsection. Figure 2 illustrates the overall architecture of the proposed GCN-DDPG hybrid model.

4.4. Proposed Reward Function

To enhance the training of the reinforcement agent, a customized reward function is designed that not only accounts for the correctness of each node's classification but also reinforces consistent predictions among neighboring nodes, as shown in Figure 3. This aligns with the concept of promoting structural stability between adjacent nodes in the graph.

Specifically, if two connected nodes are both correctly classified, an additional reward of 0.5 units is assigned. This mechanism encourages the model to focus not only on improving individual node-level accuracy but also on maintaining coherent predictions within the local graph structure.

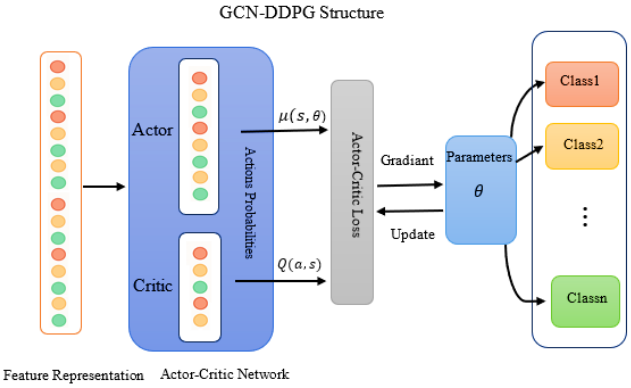


Figure 2. **Architecture of the proposed GCN-DDPG model, which combines node classification and value estimation**

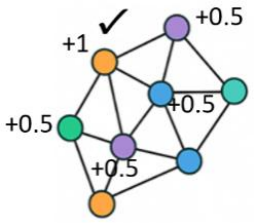


Figure 3. **The proposed reward function enhances both node-level accuracy and local structural consistency by assigning an extra reward when neighboring nodes are correctly classified**

In the proposed GCN+DDPG hybrid model, we define a reward signal $0 \le r_i \le 1$ based on the correctness of node classification. To incorporate local graph structure into the reward signal, we propose a neighborhood-aware reward function:

$$r_i = \begin{cases} 1 & \hat{y}_i = y_i \\ 0 \le r_i \le 1 & \hat{y}_i \neq y_i \text{ and } \exists j \in N(i) \text{ such that } \hat{y}_i = y_j \\ 0 & \text{otherwiser} \end{cases} \quad (12)$$

where $N(i)$ is the set of 1-hop neighbors of node i and y_i is the true label of node i. This reward encourages learning that propagates accuracy to local neighborhoods, not just the central node. In the following, the appropriate reward values for each dataset are specified in Table 2.

Table 2. **Reward Assignment for Different Datasets**

Dateset	Pubmed	Citeseer	Cora	Facebook Page	Twitch
Reward	0.25	0.5	0.25	0.5	0.25

To prevent label leakage, the reward is computed exclusively over nodes in the training set. No label information from validation or test nodes is used either directly or indirectly in reward computation. Neighbor-based reward components are restricted to neighbors that are also in the training mask.

4.5. Cost Function Calculation in Combined Training

The proposed model is trained by combining two cost functions. The first cost function is the GCN classification loss, which is used for the node classification task and matches the nodes' true labels with the predicted probability distributions (using Cross-Entropy). The second cost function is related to reinforcement learning. It consists of two components: the policy loss, computed using the policy gradient method, and the value loss, which aligns the estimated value with the actual reward. The final cost function formula is as follows:

$$combined_{loss} = loss_{GCN} + loss_{DDPG} \quad (13)$$

5. Experiments

Several benchmark datasets are employed to evaluate the performance of graph convolutional and reinforcement learning models. The datasets include diverse graph datasets such as Twitch-Gamers, Cora, PubMed, Citeseer, and Facebook, each widely used in graph learning and social network analysis with distinct characteristics. Twitch-Gamers captures social interactions among gamers on Twitch, featuring high-dimensional node attributes and complex graph structures. Cora contains scientific publications with citation links and textual features, commonly used in text classification. PubMed consists of medical research papers with citation networks and textual data, useful for biomedical literature modeling. Citeseer models citation relationships between scientific papers and their features. Facebook represents user connections and profiles in a social network and is often used for social behavior prediction. These datasets provide diverse challenges for graph representation learning. Detailed dataset information is summarized in Table 3.

Table 3. Characteristics of 5 benchmark datasets

Dataset	Nodes	Edges	Features	Class
Cora	2,708	5,429	1,433	7
Citeseer	3,327	4,732	3,703	6
Pubmed	19,717	44,338	500	3
Twitch-Gamers	10,000	186,672	2,003	2
Facebook	22,470	171,002	2,091	2

To ensure reproducibility, the random seed is fixed to 42 during training. Each experiment is repeated 5 times, and the reported results are the mean and standard deviation across these runs. The GCN model consists of two GCN convolution layers with a hidden dimension of 128, followed by a normalization layer and a dropout rate of 0.3. The Adam optimizer is used for both the GCN and the reinforcement learning components. The learning rate is set to 0.01 for both the GCN and the Actor–Critic networks, and a weight decay of 1×10^{-4} is applied to the GCN parameters. The model is trained for 200 epochs. Input features are normalized using a standard scaler, and feature selection is performed based on random forest importance scores, retaining the top 1000 features.

5.1. Performance Evaluation

In each run, the model was evaluated on the test set, and the final accuracy was recorded. After completing five runs, the mean and standard deviation of the accuracies were calculated and reported, indicating the stability of the proposed model's performance against random fluctuations. For the experiments, 80% of the nodes were allocated to training, and the remaining 20% to testing. This was implemented by defining Boolean masks within the data structure for use in the subsequent training and evaluation phases.

Based on the experiments conducted and the results summarized in Table 4, it can be concluded that combining graph convolutional models with advanced reinforcement learning techniques, such as DDPG, can significantly improve classification accuracy compared to baseline models. Furthermore, incorporating deep learning facilitators such as LSTM led to increased stability and accuracy in the A3C model. On the other hand, methods like PPO, originally designed for discrete action spaces, have demonstrated lower performance when applied to graph-structured data and node classification tasks.

These findings highlight the critical importance of selecting an appropriate reinforcement learning algorithm alongside a suitable architecture tailored to graph data. At the end of each independent run, the model's accuracy on the test set was computed and recorded. To enable statistical analysis, the mean accuracy and standard deviation were calculated over five repeated runs. Experimental results demonstrate that the hybrid GCN-DDPG model achieves both competitive accuracy and strong robustness to data fluctuations, highlighting the strength of the proposed design.

Table 4. Mean and standard deviation of Accuracy criterion (%)

Models	Twitch	FacebookPagePage	Cora	Citeseer	Pubmed
GCNA3C with LSTM	63.07±1.32	61.51±8.42	77.14±6.33	66.92±5.59	86.36±0.56
GCNA3C Without LSTM	56.70±1.15	93.89±0.21	71.98±3.88	60.22±2.14	77.78±1.83
GCNPPO	48.88±5.50	20.36±6.37	11.54±2.11	17.04±0.96	40.62±0.16
GCNDDPG with feature selection	**67.15±0.74**	94.56±0.20	**87.84±1.49**	**72. 42±2.10**	**87.15±0.12**
GCNDDPG	67.07±0.84	**94.69±0.23**	84.74±1.98	63.46±0.60	86.78±0.11
GCNREINFORCE	60.56±0.41	67.97±1.1 1	54.76±1.85	43.82±1.81	79.90±1.18
GCN	61.58±0.55	93.35±0.11	78.78±0.55	61.58±0.55	75.24±0.77
GCN with feature selection	59.24±0.22	93.37±0.15	79.34±0.37	65.98±0.50	74.88±0.92
GAT	60.41±0.60	93.89±0.22	76.98±3.03	66.34±1.34	75.04±1.52

As shown in Table 4, the GCN-DDPG model consistently outperforms the baseline GCN across all datasets. The most significant improvement is observed on the PubMed dataset with a relative increase of +11.91%, followed by notable gains of +9.06% on Cora and +5.57% on Twitch, indicating the strong potential of combining GCN with the DDPG reinforcement learning agent for node classification tasks. The GCN-A3C model with LSTM also shows improvements on several datasets, particularly achieving a significant +11.12% gain on PubMed and +5.34% on Citeseer, demonstrating that the addition of LSTM contributes to greater stability and accuracy. In contrast, its LSTM-free counterpart underperforms on most datasets, showing negative gains except for a modest +0.54% on FacebookPagePage and +2.54% on PubMed.

The experimental results demonstrate that the proposed GCN-DDPG hybrid model consistently outperforms the baseline GCN and other RL-based methods across multiple datasets, achieving both higher accuracy and robust performance on large graphs such as Facebook and PubMed. Error analysis indicates that misclassifications primarily occur on nodes with sparse local neighborhoods or ambiguous features, highlighting the model's ability to balance local and global information while revealing limitations in graphs with weak local structures. The reward function, which combines prediction correctness with neighbor consistency, further stabilizes learning and promotes convergence. Overall, these findings emphasize that

continuous-control RL algorithms, such as DDPG, are more effective than discrete-action methods, such as PPO, for node classification on graph-structured data. That careful alignment of network architecture and reward design is crucial for achieving scalable, high-fidelity graph representation learning.

Although random forest-based feature selection may slightly reduce the performance of a standalone GCN, as shown in Table 4, this occurs because GCNs inherently aggregate neighborhood information and can learn useful feature representations even when redundant attributes are present. Therefore, removing certain features during preprocessing may accidentally discard information that becomes important after graph aggregation. However, in the proposed GCN-DDPG framework, the random forest module serves a different purpose. It reduces noisy and redundant attributes, simplifies the state space for the DDPG agent, and stabilizes the overall training process. The ablation results demonstrate that incorporating random forest consistently improves the performance of the integrated model (e.g., 72.42 vs 63.46 on Citeseer and 87.15 vs 86.78 on PubMed).

The GAT model shows competitive performance only on the FacebookPagePage dataset, with a modest +0.54% improvement over the baseline GCN model, but suffers from accuracy degradation on other datasets, particularly Cora (+1.80%) and Twitch (+1.17%). Since GAT Networks rely on learnable attention coefficients to weight neighboring nodes, dense graphs provide a favorable setting in which the model can effectively identify informative neighbors while suppressing less relevant ones. Moreover, social network graphs often exhibit strong homophily and community structure, meaning that connected nodes tend to share similar labels. This further enhances the effectiveness of attention-based aggregation. GAT performs comparatively well on FacebookPagePage because its dense and homophilic topology closely matches the assumptions of attention-based message passing. In contrast, its advantage becomes less pronounced on sparse, noisy, or structurally heterogeneous graphs.

In contrast, datasets such as Cora and Citeseer are significantly sparser, providing fewer neighboring signals for reliable attention estimation. PubMed contains more complex long-range citation dependencies that local attention mechanisms may not fully capture. At the same time, Twitch includes noisy and heterogeneous node features that can reduce the robustness of attention weights. Therefore, GAT performs comparatively well on FacebookPagePage because its dense, homophilic topology closely matches the assumptions underlying attention-based message passing. In contrast, its advantage becomes less pronounced on sparse, noisy, or structurally heterogeneous graphs.

The GCN-PPO model performs poorly across all datasets, with drastic reductions in accuracy relative to the baseline GCN, including -67.24% on Cora and -72.99% on FacebookPagePage, indicating that PPO, which is more suited to discrete action spaces, is not effective for node classification in graph-structured data. Although PPO can be adapted to continuous action spaces, its canonical formulation is based on stochastic policy optimization using a clipped surrogate objective, which is generally more effective in relatively low-dimensional settings with clearly separable actions. While the clipping mechanism improves numerical stability and prevents excessively large policy updates, it may also limit the policy's responsiveness in heterogeneous graph domains, where optimal decisions vary significantly across local topological regions. In the proposed framework, decision-making is driven by continuous node embeddings learned by the GCN, where class boundaries can be highly sensitive to subtle variations in feature distributions and neighborhood connectivity patterns. Under such conditions, conservative stochastic updates may fail to exploit the geometric information encoded in the embedding space fully, leading to slower convergence and reduced classification accuracy compared with deterministic continuous-control methods such as DDPG.

The results for GCN reveal that feature selection does not affect all datasets uniformly. It improves accuracy on datasets such as Cora and Citeseer, where redundant attributes are substantial. Still, it slightly reduces performance on Twitch and PubMed, suggesting that some removed features may carry information that is useful only through graph propagation. This behavior is expected, since random forest measures marginal feature importance independently of the graph structure, whereas GCNs can exploit structurally dependent patterns.

Overall, the results confirm that the GCN-DDPG model is the most effective hybrid approach, consistently achieving significant gains in accuracy over the plain GCN. Furthermore, incorporating an LSTM into the GCN-A3C architecture improves the model's robustness and precision. On the other hand, reinforcement learning methods such as PPO and, to some extent, REINFORCE, lead to substantial performance drops, highlighting the critical importance of selecting appropriate RL algorithms for graph-based classification tasks.

In our framework, the node-selection process is naturally expressed as a continuous scoring and ranking problem rather than a discrete categorical action. For this reason, we employ DDPG, whose deterministic policy provides lower gradient variance and improved sample efficiency compared to on-policy stochastic methods such as A3C. Moreover, the off-policy replay mechanism of DDPG enhances training stability, especially in large graphs with thousands of candidate nodes. While A3C performs competitively on PubMed, DDPG offers more consistent performance across datasets, making it a better fit for our hybrid GCN-based model. Figure 4 presents the mean and standard deviation of the accuracy criterion.

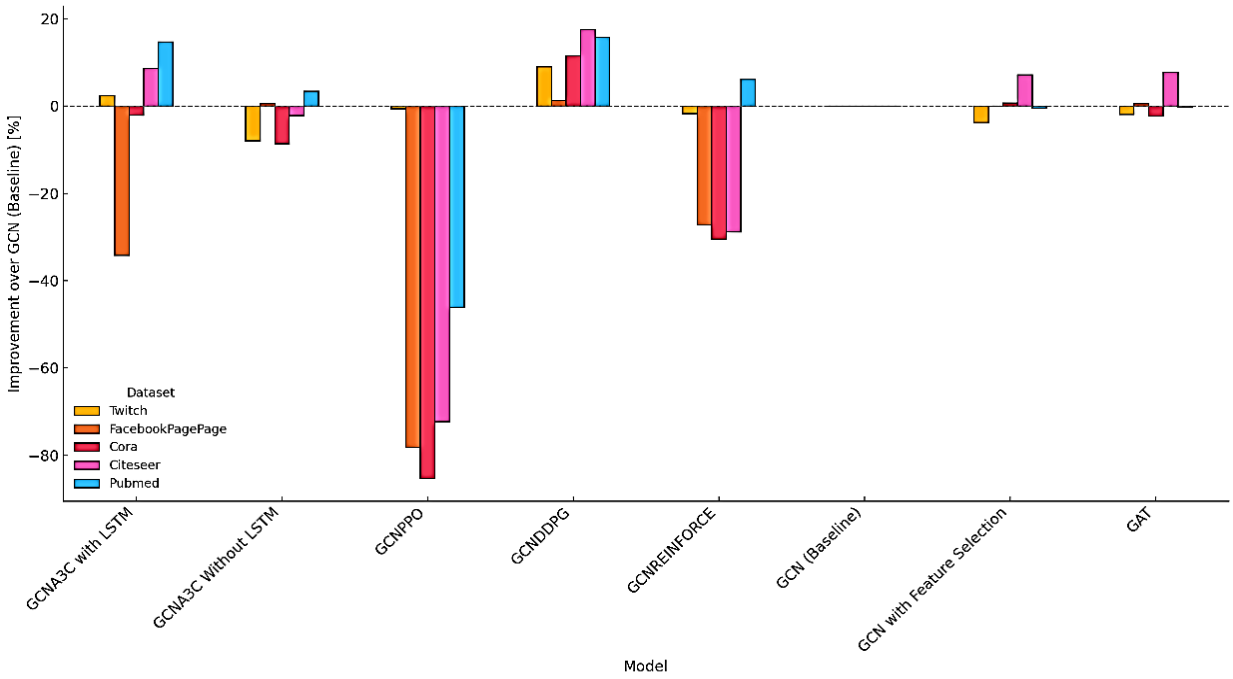


Figure 4. Comparison of model performance across datasets

6. Analysis of the Impact of Layer Number and Hidden Feature Dimension

An analysis was conducted to assess the influence of layer depth and hidden feature dimensionality on classification accuracy across the Cora, Citeseer, and PubMed datasets. For the Cora dataset, the best performance was achieved with a 2-layer architecture and a hidden dimension of 128. Single-layer architectures exhibited inferior performance, while increasing the number of layers or the hidden dimension beyond this point led to a degradation in accuracy, which can be attributed to oversmoothing. For the Citeseer dataset, a 2-layer structure with a hidden dimension of 64 achieved the highest accuracy. Deeper architectures or larger hidden dimensions resulted in reduced performance.

Similarly, for the PubMed dataset, the highest accuracy was observed with a 2-layer model and a hidden dimension of 128. Increasing either the depth or the hidden feature dimension reduced accuracy. While oversmoothing is often cited as a reason for performance degradation in deeper GCN architectures, it does not fully account for the behavior observed on the PubMed dataset. In large-scale graphs such as PubMed, oversquashing significantly degrades performance, as information from distant nodes must be compressed into fixed-size embeddings during multi-hop propagation, thereby limiting the preservation of long-range dependencies.

Additionally, repeated transformations across multiple layers may lead to feature compression, which can cause the loss of discriminative information, particularly in datasets with high-dimensional, sparse features such as PubMed. Therefore, these findings highlight that both excessive network depth and overly large hidden dimensions may negatively impact performance, likely due to oversmoothing, oversquashing, and feature compression effects.

7. Time Complexity Analysis and Model Limitations

The time and computational complexity of the graph convolutional network arise from performing convolution operations on the graph structure and from matrix multiplication between node feature and weight matrices at each graph layer. As described by Kipf and Welling [7], the graph convolution operation at layer l is computed as:

$$H^{l+1} = \sigma(\hat{A}H^{l}W^{l}) \tag{14}$$

where $\hat{A}$ denotes the normalized adjacency matrix of size $N \times N$, $H^{l}$is the node feature matrix at layer l with dimensions $N \cdot F$, N is the number of nodes, F is the number of features per node, and E is the number of edges. Given the sparsity of $\hat{A}$, the complexity of the sparse matrix multiplication is approximately $O(E \cdot F)$. Additionally, the dense matrix multiplication $H^{l}W^{l}$ contributes $O(N \cdot F^2)$. Therefore, the total time complexity of a GCN layer can be approximated as $O(N \cdot F^2 + E.F)$. This complexity grows particularly with an increase in the number of edges. In the hybrid model combining GCN and DDPG, the overall complexity includes not only the GCN computations but also the overhead introduced by DDPG. The computational complexity of DDPG arises from both the actor and critic networks, each having a complexity of approximately $O(d \cdot h + h^2)$, where d is the input dimension and h denotes the number of neurons in the hidden layers. Due to the presence of two separate neural networks (actor and critic) and the need for continuous weight updates at every time step, the computational load of the GCN-DDPG model is significantly higher than that of GCN alone.

While the proposed GCN-DDPG framework demonstrates strong performance across several datasets, it has certain limitations. First, the use of DDPG introduces considerable computational overhead because it trains two separate neural networks (actor and critic), making the model slower than a standard GCN. Second, the model requires careful tuning of

hyperparameters such as reward scaling, learning rates, and feature selection thresholds, which may limit its scalability in fully automated systems. Third, its reliance on neighborhood consistency in the reward function may not generalize well to graphs with weak or irregular local structures.

For instance, the runtime on the Cora dataset with the GCN model is approximately 9.8332 seconds, whereas the GCN–DDPG hybrid model takes 12.6910 seconds. Similarly, on the Citeseer dataset, the GCN model runs in 11.1470 seconds, whereas the hybrid model requires 17.0962 seconds. For the larger PubMed dataset, the GCN model runs in 53.3805 seconds, while the proposed GCN–DDPG framework takes 62.3081 seconds.

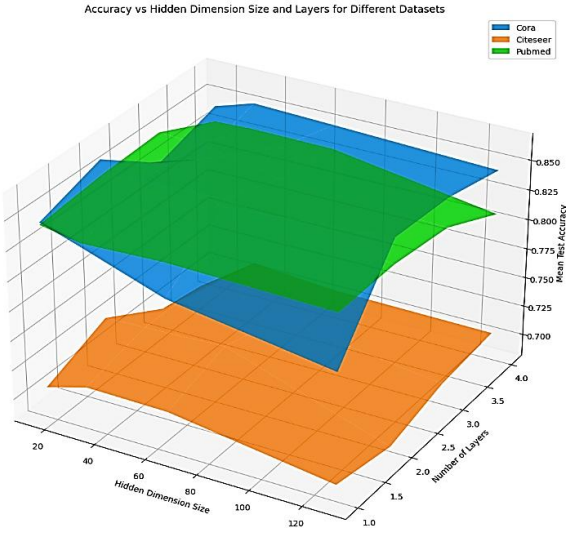


Figure 5. Accuracy vs. Layer Depth and Hidden Dimension on Cora, Citeseer, and PubMed

We also examined the memory footprint of the proposed framework across the benchmark datasets. On both Cora and Citeseer, the model maintains a memory usage of 0.16 MB. For the larger Pubmed dataset, the memory consumption increases to 17.42 MB. These results suggest that the proposed GCN–DDPG framework remains highly memory-efficient on smaller citation networks, while the increase observed on Pubmed is expected due to its larger graph scale and higher feature dimensionality. Overall, the observed memory consumption demonstrates the practicality of the proposed method for graph-based node classification tasks.

Although the current implementation follows a full-batch training strategy suitable for citation-scale networks, the proposed framework is independent of the specific GCN training paradigm. Therefore, it can be extended to large-scale graphs with millions of nodes by incorporating sampling-based techniques. Since the RL component operates on node embeddings generated by the GCN backbone and does not require additional global graph traversal, it naturally inherits the scalability properties of the underlying sampling strategy.

8. Conclusion

The integration of GCNs with DRL methods proved effective for node classification in complex graph-structured data. Among the evaluated hybrid models, the GCN-DDPG framework consistently outperformed baseline GCN and other GCN-RL combinations across multiple datasets in terms of both accuracy and robustness. This suggests that continuous-control RL algorithms, such as DDPG, are better suited to capturing fine-grained interactions in graph domains than discrete-action methods, such as PPO.

Furthermore, architectural analysis revealed that moderate-depth GCNs with carefully tuned hidden dimensions provide an optimal trade-off between representation power and oversmoothing. Although the hybrid approach introduces additional computational overhead, the observed gains in predictive performance justify its deployment for tasks that require high-fidelity graph representation learning. These findings underscore the importance of architectural alignment between GCNs and RL agents when designing scalable, high-performance graph learning systems.

9. Statements & Declarations

9.1. Author Contributions

Fatemeh Fathi Nezhad was responsible for conceptualization, methodology, software, validation, formal analysis, investigation, resources, data curation, writing - original draft preparation, writing - review and editing, visualization, supervision, and project administration. Funding acquisition was not applicable. The author has read and approved the final version of the manuscript.

9.2. Data Availability Statement

Data available in a publicly accessible repository that does not issue DOIs: Publicly available datasets were analyzed in this study. This data can be found here: www.kaggle.com/datasets.

9.3. Funding

This research received no external funding.

9.4. Acknowledgments

Not applicable.

9.5. Declaration of Generative AI and AI-assisted Technologies in the Writing Process

During the preparation of this manuscript, the author used ChatGPT (OpenAI) solely for language editing, grammar checking, and improving the clarity and readability of the text. Following the use of this tool, the author carefully reviewed and revised the content as necessary and takes full responsibility for the accuracy, integrity, and final content of the published article.

9.6. Ethics Approval

Not applicable. This study involved numerical analysis, simulation, and engineering modeling and did not involve human participants, animals, or human biological materials.

9.7. Informed Consent

Not applicable.

9.8. Consent for Publication

Not applicable. This study does not contain any personal data or identifiable information requiring consent for publication.

9.9. Conflicts of Interest

The author declares no conflict of interest.

10. References

[1] K. C. Chuang, P. S. Cheng, Y. H. Tsai, and M. H. Tsai, "Establishing a GRU-GCN coordination-based prediction model for miRNA-disease associations," BMC Genomic Data, vol. 26, no. 1, p. 4, 2025, doi:10.1186/s12863-024-01293-z.

[2] X. Wang, Z. Zhang, G. Shen, S. Lai, Y. Chen, and S. Zhu, "Multi-view knowledge graph convolutional networks for recommendation," Appl. Soft Comput., vol. 169, p. 112633, 2025, doi:10.1016/j.asoc.2024.112633.

[3] J. Liu, S. Chen, and L. Shen, "A comprehensive survey on graph neural network accelerators," Front. Comput. Sci., vol. 19, no. 2, p. 192104, 2025, doi:10.1007/s11704-023-3307-2.

[4] L. Deng, X. Zhang, L. J. Slater, H. Liu, and S. Tao, "Integrating Euclidean and non-Euclidean spatial information for deep learning-based spatiotemporal hydrological simulation," J. Hydrol., vol. 638, p. 131438, 2024, doi:10.1016/j.jhydrol.2024.131438.

[5] Q. Dang, Q. Liu, S. Yang, and X. He, "Data-Driven Evolutionary Algorithm Based on Inductive Graph Neural Networks for Multimodal Multiobjective Optimization," IEEE Trans. Evol. Comput., vol. 30, no. 1, pp. 186–198, 2026, doi:10.1109/TEVC.2025.3541046.

[6] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Message Passing Neural Networks," in Lecture Notes in Physics, vol. 968, 2020, pp. 199–214. doi:10.1007/978-3-030-40245-7_10.

[7] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in 5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings, 2017.

[8] S. Ghode and M. Digalwar, "Enhanced Electric Vehicle charging strategy through Graph Convolutional Networks integrated with deep reinforcement learning," Int. J. Inf. Technol., vol. 17, no. 7, pp. 4217–4229, 2025, doi:10.1007/s41870-024-02394-9.

[9] T. Cui, X. Yang, F. Jia, J. Jin, Y. Ye, and R. Bai, "Mobile robot sequential decision making using a deep reinforcement learning hyper-heuristic approach," Expert Systems with Applications, vol. 257, p. 124959, Dec. 2024, doi: 10.1016/j.eswa.2024.124959.

[10] V. Mnih et al., "Asynchronous methods for deep reinforcement learning," 33rd International Conference on Machine Learning, ICML, New York, United States, 19-24 June, 2016, pp. 2850–2869.

[11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput., vol. 9, no. 8, pp. 1735–1780.

[12] T. Wang, "CLE: An Integrated Framework of CNN, LSTM, and Enhanced A3C for Addressing Multi-Agent Pathfinding Challenges in Warehousing Systems," IEEE Access, vol. 12, pp. 88904–88912, 2024, doi:10.1109/access.2024.3416111.

[13] L. Tian, W. Wu, and T. Yu, "Graph Random Forest: A Graph Embedded Algorithm for Identifying Highly Connected Important Features," Biomolecules, vol. 13, no. 7, p. 1153, Jul. 2023, doi: 10.3390/biom13071153.

[14] M. Sewak, "Deep Q network (DQN), double DQN, and dueling DQN," in Deep Reinforcement Learning. Springer, Singapore, 2019, doi:10.1007/978-981-13-8285-7_8.

[15] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, "Dueling network architectures for deep reinforcement learning," in Proc. Int. Conf. Mach. Learn. (ICML), New York, United States, 19-24 June, 2016, pp. 1995–2003.

[16] J. Leng et al., "Deep Reinforcement Learning of Graph Convolutional Neural Network for Resilient Production Control of Mass Individualized Prototyping Toward Industry 5.0," IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 54, no. 11, pp. 7092–7105, Nov. 2024, doi:10.1109/tsmc.2024.3446671.

[17] P. Rokhforoz, M. Montazeri, and O. Fink, "Multi-agent reinforcement learning with graph convolutional neural networks for optimal bidding strategies of generation units in electricity markets," Expert Syst. Appl., vol. 225, p. 120010, 2023, doi:10.1016/j.eswa.2023.120010.

[18] X. Gao, X. Li, Q. Liu, Z. Li, F. Yang, and T. Luan, "Multi-Agent Decision-Making Modes in Uncertain Interactive Traffic Scenarios via Graph Convolution-Based Deep Reinforcement Learning," Sensors, vol. 22, no. 12, p. 4586, 2022, doi: 10.3390/s22124586.

[19] H. Weng, Y. Hu, M. Liang, J. Xi, and B. Yin, "Optimizing bidding strategy in electricity market based on graph convolutional neural network and deep reinforcement learning," Appl. Energy, vol. 380, p. 124978, 2025, doi:10.1016/j.apenergy.2024.124978.

[20] H. Wu, L. Tian, H. Tang, R. Li, and P. Jiao, "Graph Convolutional Reinforcement Learning-Guided Joint Trajectory Optimization and Task Offloading for Aerial Edge Computing," IEEE Transactions on Intelligent Transportation Systems, vol. 26, no. 10, pp. 17487–17498, Oct. 2025, doi:10.1109/tits.2024.3490533.

[21] G. Nurbek, "Exploring graph neural networks in reinforcement learning: A comparative study on architectures for locomotion tasks," Master's thesis, The University of Texas Rio Grande Valley, Edinburg, United States, 2024.

[22] S. Ding, W. Du, L. Ding, J. Zhang, L. Guo, and B. An, "Multiagent Reinforcement Learning with Graphical Mutual Information Maximization," IEEE Trans. Neural Networks Learn. Syst., vol. 36, no. 11, pp. 19515–19524, 2025, doi:10.1109/TNNLS.2023.3243557.

[23] Y. Guo, Q. Wu, and H. She, "A Routing Optimization Policy Using Graph Convolution Deep Reinforcement Learning," in 2023 IEEE/CIC International Conference on Communications in China, ICCC 2023, 2023, pp. 1–6. doi:10.1109/ICCC57788.2023.10233329.

[24] J. Chen and Z. Wu, "Dynamic Computation Offloading with Energy Harvesting Devices: A Graph-Based Deep Reinforcement Learning Approach," IEEE Commun. Lett., vol. 25, no. 9, pp. 2968–2972, 2021, doi:10.1109/LCOMM.2021.3094842.

[25] T. Peng et al., "GraphRARE: Reinforcement Learning Enhanced Graph Neural Network with Relative Entropy," in Proceedings - International Conference on Data Engineering, 2024, pp. 2489–2502. doi:10.1109/ICDE60146.2024.00196.

[26] M. Valko, R. Munos, B. Kveton, and T. Kocák, "Spectral bandits for smooth graph functions," in Proc. 31st Int. Conf. Mach. Learn. (ICML), Beijing, China, 21–26, Jun, 2014, pp. 46–54.

[27] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An End-to-End Deep Learning Architecture for Graph Classification," Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, no. 1, Apr. 2018, doi:10.1609/aaai.v32i1.11782.