Privacy-Preserving Graph Neural Network Inference Using Homomorphic Encryption for Secure Graph Classification

Saeed Ur Rahman[1,2], Zhao Chang[1,2], and Ke Cheng[1,2]
[1]School of Computer Science and Technology, Xidian University, Xi'an, 710071, China.
[2]Shaanxi Key Laboratory of Network and System Security, Xidian University, Xi'an, 710071, China.

Graph Neural Networks have achieved remarkable success in graph-based learning tasks, but often require access to sensitive data during inference, raising privacy concerns in domains such as healthcare, finance, and cybersecurity. Homomorphic Encryption enables computation on encrypted data; however, directly applying HE to graph neural network operations remains computationally expensive due to complex message-passing and attention mechanisms. This paper presents Hybrid-HE GNN, a privacy-preserving graph inference framework that employs parallel Graph Isomorphism Network (GIN) and Graph Attention Network (GAT) branches followed by learnable gated fusion, with embedding-level homomorphic encryption for secure downstream classification. Graph embeddings are generated in plaintext and subsequently classified under Paillier and CKKS encryption schemes using a Logistic Regression classifier, enabling secure inference while reducing computational overhead. Experimental evaluation on the MUTAG, Cora, and PROTEINS benchmark datasets demonstrates that the proposed framework preserves predictive performance under encrypted inference with only negligible deviation from plaintext execution. The Hybrid-HE GNN achieved encrypted accuracies of 87.74%, 91.10%, and 87.12% on MUTAG, Cora, and PROTEINS, respectively, outperforming baseline GNN models across the evaluated datasets. Furthermore, CKKS batched inference substantially reduced latency compared with Paillier encryption while maintaining equivalent classification performance. These results demonstrate that embedding-level homomorphic encryption provides an effective balance between privacy preservation, predictive accuracy, and computational efficiency for graph neural network inference.

Index Terms—**Graph Neural Networks (GNN), Homomorphic Encryption (HE), Privacy-Preserving Inference, Graph Classification, Secure Machine Learning.**

I. Introduction

GRAPH-structured data has become a fundamental representation paradigm for modeling complex relationships among interconnected entities across scientific, industrial, and cyber domains. Unlike conventional tabular representations, graphs explicitly preserve relational dependencies and structural interactions, enabling more expressive analysis of non-Euclidean data. Consequently, graph-based learning methods have achieved widespread adoption in applications including social network analysis, molecular property prediction, recommender systems, intelligent transportation, communication networks, cybersecurity, and healthcare analytics. Among these methods, Graph Neural Networks have emerged as one of the most effective deep learning frameworks for graph-structured data by integrating node attributes and graph topology through iterative message passing and neighborhood aggregation mechanisms [1]. Their ability to learn discriminative graph representations has led to substantial improvements in classification, prediction, and representation-learning tasks.

Despite these advantages, deploying GNNs in privacy-sensitive environments introduces significant security and confidentiality concerns. Unlike conventional machine learning pipelines, GNNs process both feature information and relational structure, creating additional attack surfaces through graph topology, neighborhood dependencies, and latent representations. Sensitive information may be exposed through intermediate embeddings, inference outputs, node relationships, or reconstructed structural patterns, increasing the risk of information leakage and regulatory non-compliance [2]. Such concerns are particularly critical in domains including healthcare, financial analytics, cybersecurity monitoring, and industrial systems, where confidential data must remain protected throughout the inference process.

To address these challenges, privacy-preserving graph learning has emerged as an active research direction. Existing approaches primarily rely on Differential Privacy, Secure Multi-Party Computation, federated learning, and Homomorphic Encryption to protect sensitive information during model training and inference [3], [4], [5]. Among these techniques, HE provides a particularly attractive capability by enabling arithmetic operations to be executed directly on encrypted data while preserving correctness after decryption. This property allows machine learning models to perform inference without revealing underlying inputs or outputs, making HE increasingly relevant for secure deployment scenarios.

Recent research has focused on addressing numerical instability, scale growth, and precision degradation during encrypted execution by optimizing polynomial evaluation and ciphertext management strategies [6]. However, integrating HE directly into GNN architectures remains challenging.

Manuscript received March 9, 2026; revised July 2, 2026. Corresponding author: Zhao Chang (email: changzhao@xidian.edu.cn).

Graph message passing requires repeated aggregation, nonlinear transformations, and neighborhood interactions that substantially increase ciphertext expansion, multiplicative depth, and encrypted execution latency. Existing encrypted GNN approaches demonstrate feasibility through polynomial approximation, ciphertext packing, and optimized encrypted execution; nevertheless, fully encrypted graph propagation continues to suffer from limited scalability and high computational overhead [7], [8], [9].

To improve deployment feasibility, recent studies increasingly adopt embedding-level encryption strategies. Rather than executing the complete graph pipeline under encryption, these approaches perform graph representation learning in plaintext and apply encryption only to compact latent representations used during downstream inference. This design significantly reduces computational cost while preserving confidentiality of sensitive outputs and intermediate representations [10], [11]. Such architectures create opportunities to combine expressive graph encoders with lightweight encrypted classifiers for practical privacy-preserving deployment.

Motivated by these observations, this work proposes a Hybrid-HE GNN framework for secure graph classification through embedding-level encrypted inference. The proposed architecture combines Graph Isomorphism Network and Graph Attention Network encoders to capture complementary structural and contextual information while maintaining compact graph representations suitable for encryption. Generated embeddings are standardized and optionally compressed before encrypted classification using Paillier and CKKS schemes. By restricting cryptographic computation to the inference stage, the framework reduces ciphertext overhead while maintaining predictive capability and deployment practicality.

Figure 1 presents the overall workflow of the proposed framework. Graph representations are first generated through hybrid graph encoding and subsequently transformed into encrypted feature spaces for secure downstream classification. This separation between representation learning and encrypted inference enables efficient deployment while avoiding the substantial computational burden associated with fully encrypted graph propagation.

The main contributions of this work are summarized as follows:

- A Hybrid-HE GNN framework that employs parallel GIN and GAT branches with learnable gated fusion to generate expressive graph embeddings suitable for privacy-preserving inference.
- An embedding-level encrypted inference pipeline integrating dimensionality reduction and homomorphic encryption to reduce ciphertext overhead and improve computational efficiency.
- A comparative evaluation of Paillier and CKKS encryption schemes under both single-instance and batched inference settings.

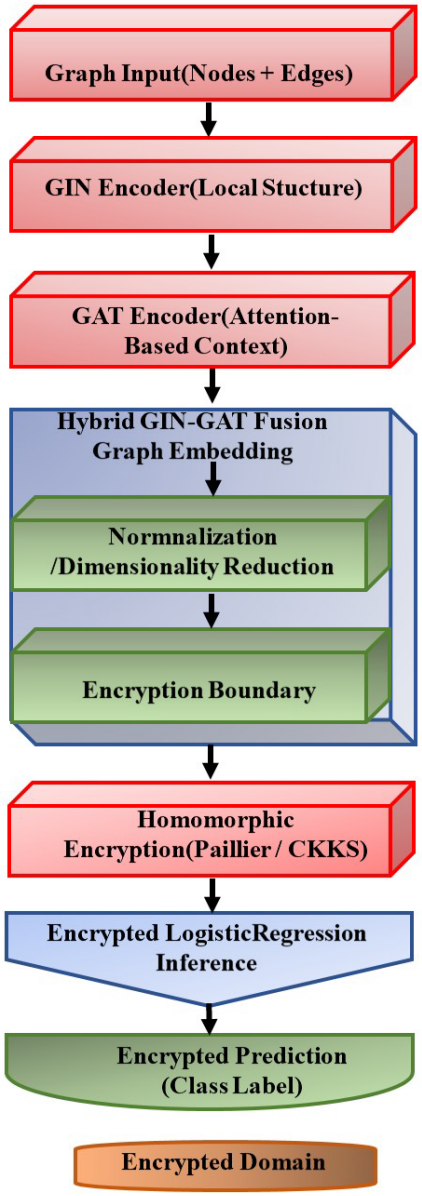

Fig. 1. Overview of the proposed framework illustrating plaintext graph representation learning followed by homomorphically encrypted inference.

- A comprehensive experimental analysis including predictive performance, latency evaluation, ablation studies, and security-oriented assessment to investigate the trade-offs between privacy preservation and encrypted inference efficiency.

II. Related Work

Research on privacy-preserving graph learning has advanced rapidly over the past five years, driven by the increasing deployment of graph-based analytics in regulated domains, including finance, healthcare, mobility, and communications. The literature is divided into several parallel but interconnected threads: privacy risks in GNNs, differential privacy for relational data, homomorphic encryption for neural inference, encrypted GNN architectures, federated learning for graphs,

and system-level tooling that enables practical encrypted computation. This section consolidates these developments with emphasis on the technical challenges, algorithmic structures, and performance constraints most relevant to homomorphic GNN inference.

A. Privacy risks and differential privacy in GNNs

Graph Neural Networks present privacy challenges that differ substantially from conventional deep learning because node attributes and graph topology are jointly exploited during representation learning. As a result, information leakage may occur through structural relationships even when explicit feature disclosure is prevented. Recent surveys report that graph-based learning remains vulnerable to attribute inference, edge reconstruction, membership inference, and embedding extraction attacks, emphasizing the need for dedicated privacy-preserving mechanisms for relational data [2], [4], [12].

Differential privacy has emerged as one of the earliest and most widely adopted approaches for mitigating privacy leakage in graph learning. Early studies introduced differential privacy mechanisms for graph learning by injecting perturbation into graph representations and training procedures to reduce information leakage. Node-level privacy preservation and private aggregation strategies were later explored to mitigate embedding exposure and structural inference attacks [13], [5]. Subsequent developments explored node-level privacy protection through learnable perturbation and privacy-preserving aggregation mechanisms to reduce embedding leakage and improve robustness under adversarial analysis [13].

More recent studies extend conventional DP formulations by introducing personalized privacy budgets and topology-aware protection strategies that adapt privacy allocation according to graph connectivity and structural heterogeneity. These methods improve the privacy utility trade-off and demonstrate better preservation of predictive performance in large and irregular graph environments [5].

B. Homomorphic encryption and encrypted neural inference

Homomorphic Encryption enables computation directly over encrypted data and has become an important foundation for privacy-preserving machine learning. Significant advances since 2020, particularly through the CKKS scheme, have enhanced support for approximate arithmetic, making encrypted neural inference increasingly practical for real-world workloads.

Recent research has focused on addressing numerical instability, scale growth, and precision degradation during encrypted execution by optimizing polynomial evaluation and ciphertext management strategies [6]. Additional studies introduced quantization-aware encrypted inference pipelines that reduce ciphertext expansion while maintaining predictive accuracy under constrained computational budgets [14].

Runtime performance has also improved substantially through advances in ciphertext packing and efficient rotation scheduling. SIMD-based parallel execution and optimized packing layouts demonstrated meaningful reductions in encrypted inference latency and memory overhead [15]. Furthermore, practical deployment has been accelerated through software ecosystems and implementation frameworks that provide encrypted tensor operators, model conversion pipelines, and HE-compatible inference interfaces [16], [17].

C. Encrypted GNNs: full-graph vs. embedding-level strategies

Applying homomorphic encryption directly to graph neural networks remains challenging because encrypted message passing introduces substantial computational overhead. Fully encrypted graph inference typically requires replacing nonlinear activation functions with polynomial approximations while controlling multiplicative depth to maintain feasible execution cost.

Early studies demonstrated the feasibility of encrypted graph learning primarily through shallow graph convolution architectures; however, extending these approaches to deeper or attention-based models remained computationally expensive due to ciphertext expansion and repeated encrypted aggregation operations [18]. Subsequent work improved scalability through parallel-packed encrypted graph computation and optimized ciphertext organization strategies for sparse graph structures [7], [19].

To address these limitations, recent research increasingly favors embedding-level encryption, where graph representation learning is performed in plaintext, and only the downstream prediction stage operates under encryption. This design substantially reduces encrypted computation while preserving model expressiveness. Existing studies have demonstrated that encrypted linear classification over graph embeddings achieves favorable privacy performance and computational efficiency [20], [10]. Additional evidence suggests that embedding compression through dimensionality reduction further lowers ciphertext cost while maintaining classification quality [21], [22].

Consequently, embedding-level encrypted inference has emerged as a practical compromise between confidentiality, computational feasibility, and predictive performance for privacy-preserving graph analytics.

D. Federated and collaborative privacy-preserving graph learning

Federated graph learning has emerged as a complementary direction for privacy-preserving graph analytics by enabling decentralized model optimization without direct sharing of graph data. Unlike centralized learning approaches, federated frameworks must address challenges associated

with distributed graph partitions, topology heterogeneity, and communication efficiency. Recent surveys summarize federated graph learning approaches and highlight challenges related to topology heterogeneity, decentralized graph partitions, and privacy preservation [23]. Subsequent studies incorporated secure aggregation and encrypted parameter exchange to strengthen protection against information leakage during collaborative optimization [24].

More recent approaches integrate homomorphic encryption, secure multi-party computation, and adaptive federated optimization to improve confidentiality while reducing communication overhead [25], [9]. Hybrid secure inference architectures further demonstrate that combining HE for arithmetic-heavy computation with MPC for interactive operations can substantially reduce execution latency and improve deployment practicality [26].

Although federated graph learning reduces direct data exposure, efficient encrypted inference remains necessary for scenarios requiring privacy-preserving deployment without distributed training dependencies.

E. Encryption-aware graph representation learning

Recent advances in privacy-preserving graph learning increasingly emphasize designing graph architectures that are compatible with encrypted computation rather than applying encryption after model execution. Conventional GNN pipelines introduce substantial encrypted execution overhead because recursive neighborhood aggregation, nonlinear activation functions, and repeated matrix operations increase multiplicative depth and ciphertext growth.

To improve practicality, recent studies have proposed encryption-aware graph processing mechanisms that reorganize graph computation under HE constraints. FicGCN introduced a graph convolution framework optimized for encrypted execution through sparse intra-ciphertext aggregation and efficient packing strategies, demonstrating that graph structural properties can significantly reduce encrypted runtime while preserving predictive performance [8]. Similarly, recent HE-oriented graph frameworks suggest that representation learning and cryptographic execution should be jointly optimized to improve scalability and reduce encrypted communication overhead [27].

These developments indicate a broader transition from fully encrypted graph processing toward privacy-preserving graph representation design. Such findings further support embedding-level encryption approaches, where graph encoders remain expressive while encrypted computation is restricted to compact latent representations.

F. Efficient Homomorphic Inference for Graph Models

Homomorphic encryption enables secure computation directly over encrypted data and provides strong confidentiality guarantees for machine learning systems. However, practical deployment remains constrained by computational latency, ciphertext growth, and high inference complexity.

Recent work has explored reducing encryption overhead through selective encryption and hybrid encrypted inference pipelines. Homomorphic Encryption-Based Federated Active Learning on Graph Convolutional Networks demonstrated that integrating encrypted learning with graph-based representation can preserve utility while reducing privacy leakage [27]. In parallel, recent encrypted GCN frameworks have shown that optimizing encrypted matrix operations and minimizing ciphertext rotations can substantially improve practical deployment performance [8].

These advances indicate that encrypting only privacy-sensitive stages of the inference pipeline offers a practical balance between security and computational efficiency. Following this principle, the proposed framework encrypts compressed graph embeddings and performs encrypted logistic regression inference under Paillier and CKKS schemes, reducing cryptographic overhead while maintaining classification performance.

G. Recent advances in privacy-preserving graph intelligence

Beyond encrypted standalone inference, recent research has expanded toward collaborative privacy-preserving graph intelligence systems capable of operating across distributed environments. These approaches integrate graph learning with federated optimization and adaptive encryption to improve scalability while maintaining confidentiality.

Recent federated graph learning frameworks combine homomorphic encryption with robust aggregation mechanisms to protect model updates and reduce communication leakage. FedGraphHE demonstrated that adaptive homomorphic encryption can preserve predictive performance while improving robustness and reducing communication cost in distributed graph learning environments [28]. Similarly, recent graph-based encrypted analytics frameworks applied homomorphic encryption to Industrial Internet of Things environments and demonstrated that privacy-preserving graph reasoning can remain practical under real-world deployment constraints [29].

Although collaborative graph learning improves data decentralization, these frameworks still introduce synchronization and communication complexity. Consequently, efficient encrypted inference remains an important direction for scenarios requiring secure deployment without distributed training.

H. System-level optimisations and HE engineering

Homomorphic inference performance is strongly dependent on system-level choices such as batching, packing, modulus scheduling, and ciphertext rotation planning. Practical optimization guidelines appear in recent surveys and tool documentation [16]. Parallel packing techniques, as explored

by Ran et al. [7], exploit SIMD parallelism for matrix-vector multiplications, while domain-specific pipelines (e.g., secure healthcare analytics) provide real-world evidence that embedding-level encrypted inference is deployable at scale [30], [3]. These findings guide the selection of HE parameters and classifier architecture in the present framework.

I. GNN Architectures and Lightweight Classifiers Relevant to Encrypted Inference

The choice of graph encoder plays a central role in determining both the fidelity of the embedding and the cost of encrypted downstream computation. Among message-passing architectures, the Graph Isomorphism Network (GIN) remains one of the most expressive frameworks for graph classification due to its injective multiset aggregation and MLP-based update functions [31]. GIN encoders consistently outperform shallower spectral models on datasets with rich structural variance, making them suitable for embedding-level encryption frameworks where high-quality latent representations are required before ciphertext expansion. Complementing GIN, Graph Attention Networks (GAT) provide adaptive neighborhood weighting through attention coefficients that modulate message importance. Multi-head attention improves the stability of the aggregation process and often yields superior performance on node-feature-rich citation networks such as Cora, CiteSeer, and PubMed. In the context of encrypted inference, GAT features are attractive because attention-driven embeddings often exhibit smoother class separation, enabling dimensionality reduction without large accuracy loss.

Once graph-level embeddings are obtained, a lightweight plaintext-to-ciphertext transition is essential for maintaining tractable encrypted computation. Linear models such as Logistic Regression are well-suited for HE-based inference because their decision functions reduce to inner products and affine transformations, both of which map efficiently to CKKS or Paillier arithmetic. Prior encrypted ML systems consistently adopt logistic or linear heads for this reason, achieving stable accuracy with minimal multiplicative depth [32], [6]. Several embedding-level GNN encryption frameworks confirm that logistic regression delivers the best latency-accuracy balance when operating on fixed-size encrypted feature vectors [20], [10]. These architectural choices, GIN for structural sensitivity, GAT for adaptive neighborhood aggregation, and logistic regression for encrypted classification, collectively enable a secure graph inference pipeline that maintains model expressivity while keeping ciphertext operations lightweight. This design aligns naturally with embedding-level encryption strategies and forms the basis for the system evaluated in the present work.

J. Synthesis and Positioning

The literature reviewed above highlights the rapid evolution of privacy-preserving graph learning while simultaneously revealing several unresolved challenges that continue to limit practical deployment. First, unlike conventional deep learning models, Graph Neural Networks expose additional privacy vulnerabilities because information leakage may arise not only from node attributes but also from graph topology and relational dependencies. Existing differential privacy mechanisms mitigate such risks through perturbation and privacy budgeting; however, maintaining an effective balance between privacy guarantees and predictive utility remains challenging, particularly for deep message-passing architectures and structurally complex graphs [2], [4], [5].

Second, substantial progress in homomorphic encryption has improved the feasibility of encrypted machine learning through advances in CKKS optimization, ciphertext packing, tensor operations, and efficient encrypted arithmetic [15], [16]. Despite these developments, fully homomorphic execution of GNN pipelines remains computationally demanding due to ciphertext expansion, multiplicative depth growth, and the repeated encrypted aggregation required by message-passing operations [18], [19], [8]. These limitations become increasingly pronounced for large-scale graphs and attention-based architectures.

Recent research, therefore, demonstrates a growing shift toward embedding-level encryption as a more practical alternative to end-to-end encrypted graph computation. By separating graph representation learning from encrypted prediction, embedding-level approaches preserve expressive latent representations while substantially reducing encryption overhead and inference latency [10], [21], [27]. However, existing studies primarily rely on single-encoder architectures and provide limited investigation into combining expressive graph representations with efficient encrypted downstream inference.

Motivated by these observations, this work proposes a Hybrid-HE GNN framework that integrates complementary graph representation capabilities through hybrid graph encoding, followed by embedding compression and privacy-preserving encrypted classification under Paillier and CKKS schemes. Unlike fully encrypted graph propagation methods or distributed privacy-preserving training frameworks, the proposed approach focuses on secure and efficient inference by restricting cryptographic computation to compact embedding spaces. This design aims to preserve classification performance while improving computational feasibility and supporting practical deployment in privacy-sensitive graph learning environments.

III. METHODOLOGY

This section presents the full design, mathematical foundations, and engineering considerations of the proposed Hybrid-HE GNN plaintext, encrypted inference framework for privacy-preserving graph classification as shown in Figure 2. The system separates computationally intensive graph encoding from the encrypted classification stage to achieve both high model expressiveness and privacy protection guarantees.

A. Threat Model

An honest-but-curious (semi-honest) adversarial model is assumed, in which the inference server correctly follows the prescribed protocol while attempting to infer sensitive information from the data observed during computation. A graph is represented as

$$G = (V, E, X) \tag{1}$$

where (V) denotes the set of nodes, (E) represents the edge set, and (X) denotes the node feature matrix. The trained graph neural network encoder $f(\cdot)$ transforms the graph into a latent representation.

$$z = f(G) \tag{2}$$

where $z \in R^d$ is the graph embedding used for downstream classification. To preserve privacy, the embedding is encrypted using a homomorphic encryption scheme before transmission to the inference server:

$$\mathbf{c} = \mathrm{Enc}_{\mathsf{HE}}(\mathbf{z}), \tag{3}$$

Here, $\mathrm{Enc}_{\mathsf{HE}}(\cdot)$ denotes encryption under either the Paillier or CKKS scheme, depending on the evaluation setting. Classification is then performed directly on encrypted embeddings. For the logistic regression classifier, the prediction function is defined as

$$\hat{y} = \sigma\left(w^T z + b\right) \tag{4}$$

where (w) and (b) represent the classifier parameters and $\sigma(\cdot)$ denotes the sigmoid activation function. Under homomorphic encryption, inference is evaluated on ciphertexts as

$$c_{\hat{y}} = \mathrm{Enc}_{\mathsf{HE}}\left(\mathbf{w}^T \mathbf{z} + b\right). \tag{5}$$

and the final prediction can only be recovered by the data owner through

$$\hat{y} = \mathrm{Dec}_{\mathsf{HE}}\left(c_{\hat{y}}\right). \tag{6}$$

where $\mathrm{Dec}(\cdot)$ denotes the decryption function.

The adversary may observe encrypted embeddings, encrypted prediction outputs, ciphertext sizes, communication patterns, execution times, and public model parameters. However, access to plaintext embeddings, prediction values, and secret decryption keys is not available. The primary objective is to prevent the disclosure of sensitive information contained within graph representations. Privacy leakage is formally defined as the information gained by an adversary regarding a sensitive variable (S) after observing the ciphertext (C):

$$L(S; C) = I(S; C) \tag{7}$$

where $I(\cdot)$ denotes mutual information. A semantically secure homomorphic encryption scheme aims to ensure

$$I(S; C) \approx 0 \tag{8}$$

indicating that ciphertext observations reveal negligible information about the underlying graph embeddings. Consequently, graph-level embeddings, intermediate classifier computations, and prediction outputs remain protected during inference.

The proposed framework provides embedding-level privacy preservation rather than fully encrypted graph processing. Graph topology (E), node features (X), and message-passing operations are processed in plaintext within a trusted environment and therefore remain outside the protection boundary of the encryption scheme. In addition, side-channel information such as ciphertext dimensions, runtime characteristics, and communication volume may disclose limited metadata regarding model configuration or embedding dimensionality. Nevertheless, such information does not reveal graph contents or prediction values under the security assumptions of the Paillier and CKKS cryptosystems. Therefore, security guarantees rely on the semantic security properties of the underlying homomorphic encryption schemes and the assumption that secret decryption keys remain inaccessible to the inference server and external adversaries.

B. Notation and Problem Formulation

The study focuses on a supervised graph classification setting where each sample is a graph $G = (V, E, X)$, with V denoting the set of nodes, E the set of edges, and $X \in R^{|V| \times d}$ the node-feature matrix. The Cora dataset is treated as a node classification benchmark, where embeddings are generated for individual nodes within a single citation graph. In contrast, MUTAG is a graph classification dataset where graph-level embeddings are obtained through pooling operations. These two settings are evaluated separately to avoid conceptual inconsistency.

$$f : \mathcal{G} \rightarrow, Y \tag{9}$$

where $y \in \{1, \ldots, C\}$ denotes the class label predicted by the Hybrid-HE GNN encoder followed by a logistic regression classifier. Let $h_v^{(k)}$ denote the node embedding of node v at layer k, and let

$$\mathbf{H}^{(K)} = \left\{ \mathbf{h}_v^{(K)} \mid v \in V \right\} \tag{10}$$

represent the final-layer node embeddings. A global pooling operator $\mathcal{P}(\cdot)$ maps these node embeddings to a graph-level vector

$$\mathbf{z} = \mathcal{P}\left(\mathbf{H}^{(K)}\right) \in R^{d_z} \tag{11}$$

This vector is then compressed, encrypted, and used for classification.

Similarly, two homomorphic encryption schemes are employed: (i) Paillier, supporting exact additive homomorphism, and (ii) CKKS, supporting approximate arithmetic for vectorized encrypted computation.

Given an embedding z, the encrypted classifier computes

$$\mathrm{Enc}(y) = \mathrm{Enc}\big(\mathbf{w}^\top \mathbf{z} + b\big) \tag{12}$$

without revealing z, w, or intermediate computations to the inference server.

For Paillier, the encrypted score is a sum of encrypted-scaled embedding values multiplied by plaintext weights, enabling binary classification for MUTAG. For CKKS, batching enables multiple values of z to be packed into a single ciphertext, facilitating efficient multiclass inference on Cora. The plaintext model parameters are stored on the client or trusted domain, while embeddings remain encrypted throughout.

All vectors appear in boldface (e.g., $\mathbf{x}$), matrices in uppercase bold (e.g., $\mathbf{W}$), and Encryption is denoted by $\mathrm{Enc}_P(\cdot)$ for Paillier and $\mathrm{Enc}_C(\cdot)$ for CKKS, while decryption is denoted by $\mathrm{Dec}_P(\cdot)$ and $\mathrm{Dec}_C(\cdot)$, respectively. Throughout this work, $\langle \cdot, \cdot \rangle$ denotes the inner product, and $\odot$ represents elementwise multiplication when applicable.

C. Graph Encoder Architecture

The Hybrid-HE GNN encoder combines the complementary strengths of Graph Isomorphism Networks (GIN) and Graph Attention Networks (GAT) to generate expressive, compact graph-level embeddings suited for subsequent homomorphic encryption. The encoder is designed around three objectives: (i) maximizing structural discriminability, (ii) preserving informative node-level interactions, and (iii) producing embeddings with dimensions appropriate for CKKS and Paillier encryption, where ciphertext size directly impacts latency and noise growth.

1) Message-Passing Framework

The encoder follows the standard message-passing paradigm in which each node representation is iteratively updated by aggregating information from its neighbors. Formally, a graph is represented as $G = (V, E)$ with node set V, edges E, and node features $X \in R^{|V| \times d}$. A generic message-passing layer can be written as:

$$h_v^{(k)} = \phi^{(k)} \left(h_v^{(k-1)},\ {}_{u \in \mathcal{N}(v)} \psi^{(k)}(h_v^{(k-1)}, h_u^{(k-1)}, e_{uv}) \right) \tag{13}$$

where $\phi^{(k)}$ is the update function, $\psi^{(k)}$ computes messages, and denotes the aggregation operator (sum, mean, or attention).

The encoder employs two parallel message-passing branches, namely a Graph Isomorphism Network (GIN) and a Graph Attention Network (GAT). Both branches receive the same input node representations and independently learn complementary structural and contextual features. Their outputs are subsequently combined through a learnable gated fusion mechanism. Thus, the GIN and GAT operations are performed in parallel rather than sequentially, and the resulting representations are fused before graph-level readout.

2) GIN Block: Expressive Structural Encoding

The GIN layer is selected for its high expressive power, approaching the discriminative capability of the Weisfeiler-Lehman (1-WL) test. The GIN update takes the form:

$$h_v^{(k)} = \mathrm{MLP}_k \left((1 + \epsilon_k) \cdot h_v^{(k-1)} + \sum_{u \in \mathcal{N}(v)} h_u^{(k-1)} \right), \tag{14}$$

where ϵ_k is a learnable scalar controlling the balance between self-features and neighborhood aggregation.

This formulation ensures a strictly injective aggregation over multisets of neighbor features, which is critical for tasks where node permutations or subtle structural changes must be distinguished. The GIN block, therefore, produces a structural backbone that captures graph topology and degree-aware neighborhood statistics.

3) GAT Block: Adaptive Attention-Based Refinement

In parallel with the GIN branch, the GAT branch learns adaptive neighborhood representations by assigning attention weights to neighboring nodes. The attention mechanism computes context-dependent coefficients that determine the relative importance of neighboring representations. The GAT attention mechanism computes unnormalized coefficients:

$$e_{uv} = \mathrm{LeakyReLU}\left(a^\top [W h_u \parallel W h_v] \right), \tag{15}$$

followed by a softmax normalization:

$$\alpha_{uv} = \frac{\exp(e_{uv})}{\sum_{k \in \mathcal{N}(v)} \exp(e_{vk})}. \tag{16}$$

The node update rule becomes:

$$h'_v = \sigma \left(\sum_{u \in \mathcal{N}(v)} \alpha_{uv} W h_u \right), \tag{17}$$

which selectively amplifies or suppresses neighbor contributions.

Multi-head attention is employed to stabilize training and capture diverse relational patterns. Formally, with M heads:

$$h'_v = \|_{m=1}^{M}\ \sigma \left(\sum_{u \in \mathcal{N}(v)} \alpha_{uv}^{(m)} W^{(m)} h_u \right). \tag{18}$$

This block introduces interpretability, making it possible to visualize edge importance (later useful for encrypted decision justification without revealing raw features).

The proposed Hybrid-HE GNN employs a parallel dual-branch encoder. Given the same input graph representation, the GIN and GAT branches independently generate structural and attention-aware node representations, respectively. A learnable gate then combines the two representations before graph-level readout and embedding compression. Therefore, the architecture should be interpreted as parallel GIN/GAT

processing followed by gated fusion, rather than as a sequential GIN-to-GAT pipeline.

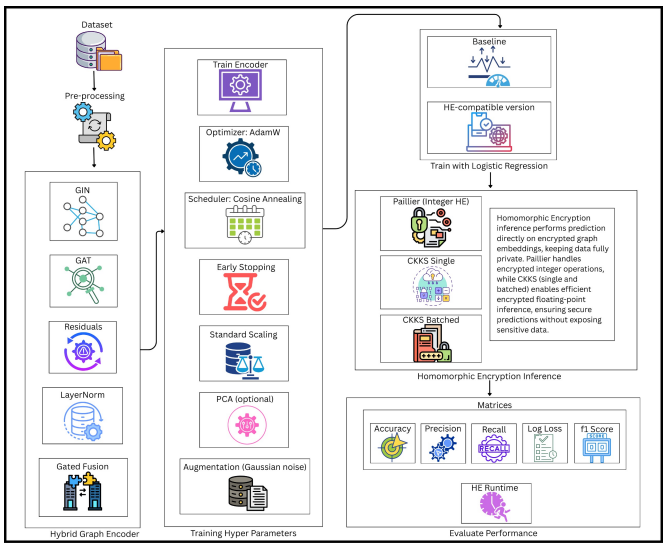

Fig. 2. Overall architecture of the proposed Hybrid-HE GNN encoder with encrypted inference using Logistic Regression under Paillier and CKKS.

4) Graph Readout and Embedding Projection

After node-level propagation, graph-level pooling is performed for graph classification datasets such as MUTAG. For node classification datasets such as Cora, node embeddings are used directly without global pooling.

$$z_G = \text{concat}\left(\sum_{v\in V} h'_v,\ \frac{1}{|V|}\sum_{v\in V} h'_v\right). \tag{19}$$

This dual pooling captures both magnitude-sensitive and distribution-sensitive information, improving downstream classification performance.

To reduce ciphertext size, a projection layer compresses z_G into d_{enc} dimensions:

$$\tilde{z}_G = P\, z_G, \quad P \in R^{d_{\text{enc}}\times d_{\text{raw}}}. \tag{20}$$

The value of d_{enc} is chosen to minimize CKKS ciphertext slots while retaining class separability, typically in the range 16 64 depending on dataset size and complexity.

5) Rationale for GIN+GAT Hybridization

The proposed encoder uses parallel GIN and GAT branches followed by gated feature fusion. The two branches are designed to capture complementary properties of graph-structured data:

- **Structural sensitivity:** The GIN branch captures fine-grained structural differences through injective neighborhood aggregation.
- **Attention-guided representation:** The GAT branch assigns adaptive importance to neighboring nodes and captures contextual relationships.
- **Adaptive feature fusion:** A learnable gate combines the GIN and GAT representations according to their relative contribution to the downstream classification task.
- **Compact homomorphic-ready embeddings:** The fused representation is subsequently pooled and projected to a compact dimensionality suitable for homomorphic encryption.

The resulting embedding serves as the fully encrypted input for the logistic regression classifier, enabling privacy-preserving inference without exposing graph features, node labels, or intermediate activations.

D. Encryption Pipeline

The encryption stage transforms the plaintext graph embeddings into encrypted vectors suitable for secure inference under homomorphic arithmetic. The pipeline consists of four tightly connected components: (i) Preprocessing and fixed-point encoding, (ii) ciphertext generation using Paillier and CKKS, (iii) encrypted linear evaluation through homomorphic logistic regression, and (iv) encrypted decision computation without information leakage.

1) Preprocessing and Fixed-Point Encoding

Let $\mathbf{h}_i \in R^d$ denote the final graph-level embedding extracted from the Hybrid-HE GNN encoder. Before encryption, embeddings undergo normalization and optional dimensionality reduction to reduce ciphertext size and improve noise stability. A standard transformation is applied:

$$\tilde{\mathbf{h}}_i = \frac{\mathbf{h}_i - \mu}{\sigma}, \tag{21}$$

where μ and σ denote dataset-level mean and standard deviation.

For CKKS, embeddings are encoded through fixed-point polynomial representation such that:

$$\mathbf{h}_i^{(\text{enc})} = \text{Encode}_\Delta(\tilde{\mathbf{h}}_i), \tag{22}$$

where Δ is the scaling factor controlling precision.

For Paillier, which supports only integer arithmetic, embeddings are multiplied by a scaling factor S and rounded:

$$\mathbf{h}_i^{(\text{int})} = \lfloor S \cdot \tilde{\mathbf{h}}_i \rfloor, \tag{23}$$

ensuring compatibility with the additive homomorphic structure.

2) Paillier Encryption of Embedding Vectors

The Paillier cryptosystem is applied when operations require additive homomorphism and exact aggregation. Each integer component $h_{i,j}^{(\text{int})}$ is encrypted as:

$$c_{i,j} = \text{Enc}_P\left(h_{i,j}^{(\text{int})}\right) = g^{h_{i,j}^{(\text{int})}} r^n \bmod n^2, \tag{24}$$

where n is the RSA modulus, $g = n+1$ following common parameterization, and r is a random integer satisfying $\gcd(r,n) = 1$. Paillier supports:

$$\text{Enc}_P(a)\,\text{Enc}_P(b) = \text{Enc}_P(a+b), \tag{25}$$

$$\mathrm{Enc}_P(a)^k = \mathrm{Enc}_P(ka). \tag{26}$$

which enables encrypted evaluation of linear models such as logistic regression without requiring decryption.

3) CKKS Encryption for Approximate Arithmetic

The CKKS scheme enables encrypted computation on floating-point vectors through approximate polynomial arithmetic. Each vector $\tilde{\mathbf{h}}_i$ is encoded into a complex polynomial $m(X)$ under the ring:

$$\mathcal{R}_q = Z_q[X]/(X^n + 1), \tag{27}$$

and encrypted using RLWE-based ciphertext generation:

$$\mathbf{c} = (c_0, c_1) = (b + a \cdot s + e, -a), \tag{28}$$

where s is the secret key, a is sampled uniformly from $\mathcal{R}_q$, b consists of the message plus scaled error, and e is a small noise term.

$$\mathbf{c}_i = \mathrm{Enc}_C\left(\tilde{\mathbf{h}}_i\right), \tag{29}$$

CKKS supports vectorized SIMD-style packing, allowing multiple embedding entries to be encrypted within a single ciphertext. This property enables efficient encrypted logistic regression and matrix-vector operations.

4) Encrypted Logistic Regression Inference

Encrypted inference applies the logistic regression classifier to the encrypted embedding. Let the plaintext model be defined by the weight vector $\mathbf{w}$ and bias b. The linear decision function is:

$$z_i = \mathbf{w}^T \mathbf{h}_i + b, \tag{30}$$

Under Paillier:

$$\mathrm{Enc}_P(z_i) = \prod_{j=1}^{d} \mathrm{Enc}_P\left(h_{i,j}^{(\mathrm{int})}\right)^{w_j^{(\mathrm{int})}} \cdot \mathrm{Enc}_P\left(b^{(\mathrm{int})}\right), \tag{31}$$

where $w_j^{(\mathrm{int})}$ and $b^{(\mathrm{int})}$ are scaled integer parameters.

Under CKKS, the encrypted inner product becomes:

$$c_{z_i} = \sum_{j=1}^{d} w_j \, c_{i,j} + b, \tag{32}$$

where $c_{i,j}$ denotes the encrypted CKKS representation of the j-th embedding component, and the plaintext weights and bias are applied through ciphertext-plaintext multiplication and addition.

Since the logistic sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$ Involves exponential operations that are not directly supported under homomorphic encryption, a low-degree polynomial approximation is employed. In particular, a cubic approximation is adopted:

$$\sigma(z) \approx a_0 + a_1 z + a_3 z^3, \tag{33}$$

where $a_0 = 0.5$, $a_1 = 0.197$, and $a_3 = -0.004$. The quadratic term is omitted ($a_2 = 0$) to limit multiplicative depth, making the approximation suitable for efficient encrypted inference while preserving sufficient classification accuracy.

5) Ciphertext Noise Management and Rescaling

CKKS ciphertexts accumulate noise from homomorphic multiplication and addition. After each multiplication, ciphertext rescaling is applied:

$$\mathbf{c} \leftarrow \mathrm{Rescale}(\mathbf{c}), \tag{34}$$

to reduce modulus q and maintain numerical precision.

Rotation keys are employed to align packed elements during dot-product operations.

6) Secure Decision Extraction

The encrypted prediction score $\mathsf{Enc}(z_i)$ is transmitted to the authorized holder of the secret key. Decryption yields:

$$\hat{z}_i = \mathrm{Dec}_{\mathsf{HE}}(c_{z_i}), \tag{35}$$

For CKKS, the authorized decryptor applies the CKKS secret key to the encrypted score c_{z_i} and obtains an approximate plaintext value $\hat{z}_i$ due to the approximate nature of CKKS arithmetic.

followed by:

$$\hat{y}_i = I\left[\sigma(\hat{z}_i) > 0.5\right], \tag{36}$$

where $I[\cdot]$ denotes the indicator function, which returns one when the condition is satisfied and zero otherwise.

Only the final class label or probability is revealed; embeddings, intermediate activations, and model computations remain encrypted throughout the entire pipeline.

7) Summary of Computational Flow

The full encryption workflow is summarized as:

1) Extract graph embedding $\mathbf{h}_i$ from GIN-GAT encoder.
2) Normalize and encode embedding into fixed-point representation.
3) Encrypt embedding through Paillier (integer arithmetic) or CKKS (approximate arithmetic).
4) Evaluate logistic regression classifier over ciphertexts.

5) Manage polynomial modulus, noise, and rescaling in CKKS.
6) Return encrypted prediction to the authorized decryptor.

Figure 1 and 2 in Section 1 and Section 3 visually depict the ciphertext flow, the transition from plaintext embeddings to encrypted vectors, and the structure of the homomorphic inference circuit.

E. Training Strategy

The Hybrid-HE GNN encoder and the subsequent encrypted classifier are trained using a staged strategy that separates representation learning from secure inference. This separation ensures that the computational burden associated with Homomorphic Encryption (HE) does not interfere with gradient-based optimization, enabling stable convergence and the extraction of high-quality graph embeddings suitable for encrypted downstream classification.

1) Dataset Preparation and Graph Preprocessing

Each dataset is first converted into its graph-structured form $\mathcal{G} = (V, E, X)$, where V denotes nodes, E edges, and $X \in R^{|V|\times d}$ node attributes. The MUTAG and PROTEINS datasets is treated as a graph classification benchmark, where each graph corresponds to an independent molecular structure. In contrast, for node classification datasets such as Cora, the graph is treated as a single large citation network where labels are associated with individual nodes, and embeddings are learned at the node level without decomposing the graph into multiple samples. In contrast, for graph classification datasets such as MUTAG and PROTEINS, each sample corresponds to an independent graph with a single label, requiring graph-level embedding generation through pooling operations. These settings are handled separately throughout the experiments to avoid conceptual inconsistency between graph-level and node-level tasks.

Graphs may vary in the number of nodes and edges, creating heterogeneity in graph sizes for graph classification tasks. In contrast, node classification operates on a fixed graph structure with varying neighborhood connectivity. The Preprocessing pipeline includes:

- **Normalization of node features**: $X' = \frac{X-\mu}{\sigma}$, ensuring consistent scaling.
- **Adjacency augmentation**: A self-loop is added to every node so the encoder can aggregate local information:

$$\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}_{|V|} \tag{37}$$

- **Degree-normalized adjacency**:

$$\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\hat{\mathbf{A}}\mathbf{D}^{-\frac{1}{2}} \tag{38}$$

- **Label smoothing**: Applied for multi-class datasets to reduce overfitting.

Graph batches are formed using standard graph batching operators, enabling parallel training of variable-sized graphs.

2) Forward Pass in the Hybrid-HE GNN Encoder

The encoder consists of two parallel branches: a Graph Isomorphism Network branch and a Graph Attention Network branch. Both branches process the same input node representations and are optimized jointly during training. Their outputs are subsequently combined using a learnable gated fusion mechanism. The training phase therefore optimizes the parameters of both branches together with the fusion-gate parameter:

$$\Theta = \Theta_{\text{GIN}} \cup \Theta_{\text{GAT}} \cup \{\gamma\}. \tag{39}$$

a) GIN Propagation.: A GIN layer produces:

$$h^{(k)}(v) = \text{MLP}^{(k)}\left((1+\epsilon^{(k)})\cdot h^{(k-1)}(v) + \sum_{u\in\mathcal{N}(v)} h^{(k-1)}(u)\right)$$

The learnable scalar $\epsilon^{(k)}$ enables fine-grained control over identity mapping influence.

b) GAT Propagation.: A GAT layer computes attention coefficients:

$$e_{vu} = \text{LeakyReLU}\big(\mathbf{a}^\top [\mathbf{W}\mathbf{h}(v) \parallel \mathbf{W}\mathbf{h}(u)]\big) \tag{40}$$

Normalized using softmax:

$$\alpha_{vu} = \frac{\exp(e_{vu})}{\sum_{u'\in\mathcal{N}(v)} \exp(e_{vu'})} \tag{41}$$

And produces:

$$\mathbf{h}_{\text{GAT}}(v) = \sum_{u\in\mathcal{N}(v)} \alpha_{vu}\mathbf{W}\mathbf{h}(u) \tag{42}$$

c) Fusion of GIN and GAT Outputs.: The two branches operate in parallel and produce complementary node representations. Their outputs are combined using a learnable fusion gate:

$$\mathbf{h}^{(v)}_{\text{fusion}} = g\,\mathbf{h}^{(v)}_{\text{GIN}} + (1-g)\,\mathbf{h}^{(v)}_{\text{GAT}}, \tag{43}$$

where

$$g = \sigma(\gamma), \qquad g \in [0,1], \tag{44}$$

and γ is a learnable scalar. A value of g closer to one increases the contribution of the GIN representation, whereas a value closer to zero increases the contribution of the GAT representation.

3) Graph-Level Embedding and Readout

Graph classification (e.g., MUTAG) requires a global representation, whereas node classification (e.g., Cora) operates directly on node-level embeddings without global aggregation.

$$\mathbf{z}_{\mathcal{G}} = \text{READOUT}\left(\{\mathbf{h}_{\text{fusion}}(v) \mid v \in V\}\right) \tag{45}$$

Two readout functions are applied:

$$\mathbf{z}_{\text{mean}} = \frac{1}{|V|}\sum_{v\in V} \mathbf{h}_{\text{fusion}}(v) \tag{46}$$

$$\mathbf{z}_{\text{mean}} = \frac{1}{|V|}\sum_{v\in V} \mathbf{h}_{\text{fusion}}(v) \tag{47}$$

The final embedding is concatenated:

$$\mathbf{z} = [\mathbf{z}_{\text{sum}} \parallel \mathbf{z}_{\text{mean}}] \tag{48}$$

A dense projection layer reduces dimensionality to a compact vector suitable for HE:

$$\hat{\mathbf{z}} = \mathbf{W}_p \mathbf{z} + \mathbf{b}_p \tag{49}$$

4) Loss Function and Optimization

Training minimizes cross-entropy loss:

$$\mathcal{L}_{\text{CE}} = -\sum_{c=1}^{C} y_c \log \hat{y}_c \tag{50}$$

Regularization terms include:

$$\mathcal{L}_{\text{reg}} = \beta_1 \|\Theta\|_2^2 + \beta_2 \|\hat{\mathbf{z}}\|_2^2 \tag{51}$$

The combined training objective is:

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{reg}} \tag{52}$$

Optimization uses Adam with:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \frac{\mathbf{m}_t/(1-\beta_1^t)}{\sqrt{\mathbf{v}_t/(1-\beta_2^t)} + \epsilon} \tag{53}$$

Warm-up scheduling and early-stopping criteria stabilize training.

5) Embedding Stabilization for Homomorphic Encryption

Since encrypted inference requires low-noise, bounded-magnitude vectors, an embedding stabilization phase is incorporated:

- **L2-normalization:** $\tilde{z} = \frac{\hat{z}}{\|\hat{z}\|_2}$
- **Quantization-aware training:** Embeddings are softly rounded:

$$\tilde{\mathbf{z}}_q = \frac{\text{round}\,(\alpha \tilde{\mathbf{z}})}{\alpha} \tag{54}$$

- **Principal Component Reduction:**

$$\tilde{\mathbf{z}}_{\text{PCA}} = \mathbf{P}_k^\top \tilde{\mathbf{z}} \tag{55}$$

Where k is chosen to minimize CKKS ciphertext growth.

These adjustments significantly reduce ciphertext noise growth and improve the success rate of encrypted multiplication and rotation operations.

6) Logistic Regression Training for Secure Classification

The encrypted inference pipeline uses Logistic Regression as the encrypted classifier due to its linear structure, minimal multiplicative depth, and compatibility with Paillier and CKKS.

During training, plaintext embeddings $\tilde{z}_{PCA}$ optimize:

$$\hat{y} = \sigma\left(\mathbf{w}^\top \tilde{\mathbf{z}}_{\text{PCA}} + b\right) \tag{56}$$

where:

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{57}$$

The logistic regression objective is:

$$\mathcal{L}_{\text{LR}} = -y \log(\hat{y}) - (1-y)\log(1-\hat{y}) \tag{58}$$

Since the sigmoid function is not homomorphically compatible, encrypted inference later replaces:

$$\sigma(x) \approx \text{sign}(x) \quad \text{or a polynomial approximation.}$$

The classifier parameters (w, b) are exported in plaintext and loaded into the HE pipeline for encrypted operations.

7) Model Freezing Before Encryption

After training converges:

- all encoder weights are frozen,
- PCA projection and normalization parameters are stored,
- final embeddings for inference use:

$$\tilde{\mathbf{z}}_{\text{enc}} = \mathbf{P}_k^\top \left(\frac{\mathbf{W}_p \mathbf{z} + \mathbf{b}_p}{\|\mathbf{W}_p \mathbf{z} + \mathbf{b}_p\|_2} \right) \tag{59}$$

This ensures deterministic embedding generation during encrypted inference, eliminating noise divergence issues.

8) Inference Preparation for Encryption

For encrypted inference, each embedding must satisfy:

$$|\tilde{z}_{\text{enc},i}| < 1 \tag{60}$$

$$\text{Var}(\tilde{\mathbf{z}}_{\text{enc}}) \approx 0.1 \text{ to } 0.3 \tag{61}$$

These constraints prevent CKKS ciphertext saturation and ensure that Paillier addition remains numerically stable. Embeddings violating these thresholds are automatically rescaled and clipped before encryption.

IV. Proposed embedding-levelHybrid-HE GNN HE Algorithm

The overall algorithm integrates graph representation learning, statistical classification, and homomorphic encryption into a unified privacy-preserving workflow. Algorithm 1 outlines the embedding-level sequence implemented in the system. The process begins by loading the dataset and constructing stratified folds to ensure balanced label distribution across training, validation, and testing partitions. For each fold, a Hybrid-HE GNN encoder based on GIN and GAT layers is trained using AdamW optimization with cosine-annealing scheduling, early stopping, optional DropEdge, and label smoothing to stabilize learning on small graph datasets. Once trained, the encoder transforms each graph into a fixed-length embedding through node-level processing, gated GIN–GAT fusion, and global pooling. These embeddings are standardized and optionally passed through PCA to reduce dimensionality before encryption, thereby decreasing ciphertext size and improving homomorphic performance. Logistic Regression is used as the downstream classifier because its linear decision boundary maps cleanly onto encrypted arithmetic. To improve classifier robustness, the training data is augmented by injecting small Gaussian

TABLE I
HYPERPARAMETERS USED IN THE HYBRID-HE GNN, EMBEDDING PREPARATION, AND ENCRYPTED INFERENCE PIPELINE

Component	Parameter	Value / Description
Hybrid-HE GNN Encoder	Hidden dimension	128
	Embedding dimension	256
	Attention heads	4
	Dropout	0.40
	Attention dropout	0.00
	Residual connections	Enabled on both GIN and GAT branches
Training Configuration	Epochs	50
	Optimizer	AdamW
	Learning rate	3×10^{-4}
	Weight decay	1×10^{-4}
	Scheduler	CosineAnnealingLR
	Early stopping	Patience = 10
	DropEdge probability	0.0 (configurable)
Embedding Preparation	Standardization	StandardScaler (fitted on train+val embeddings)
	PCA compression	Enabled, target dimension = 32
	Logistic regression space	PCA-transformed embeddings
	LR data augmentation	Gaussian noise ($\sigma = 10^{-2}$), 2 copies
	Final HE embedding size	32-D vector
Logistic Regression Classifier	Solver	LBFGS, max_iter = 5000
	Class weighting	Balanced
	Output type	Probabilities for binary classification
	Training dataset	Augmented train + validation embeddings
Paillier HE Settings	Key length	2048 bits
	Integer scaling factor	5000
	Inference mode	Per-sample encrypted logistic regression
	Complexity	Encrypted dot product + encrypted bias addition
CKKS HE Settings	Polynomial modulus degree	8192
	Global scale	2^{40}
	Coefficient modulus sizes	$\{60, 40, 40, 60\}$
	Inference modes	Single-sample + batched (size = 8)
	Rotation keys	Enabled for batched dot products
Cross-Validation Protocol	Splits	Stratified 5-fold cross-validation
	Evaluation	Mean ± Std across folds
	Metrics	Accuracy, precision, recall, F1, ROC-AUC, log-loss
	Test flow	Plaintext vs Paillier vs CKKS (single+batched)

perturbations into the embeddings, thereby enhancing the stability of the decision margin.

After training a plaintext classifier, two versions of the classifier weights are produced: the full-dimensional model for baseline evaluation and the PCA-compressed model for encrypted inference. The encrypted inference procedure is executed under three schemes: Paillier (integer-based additive homomorphism), CKKS single-sample inference (floating-point approximate HE), and CKKS batch inference that leverages ciphertext packing for vectorized dot products. Each encrypted prediction computes the encrypted weighted sum of embeddings and the classifier parameters, followed by decryption and numerical conversion to logits for probability estimation. The pipeline outputs accuracy, precision, recall, F1-score, ROC AUC, log-loss, and wall-clock latency for each encryption method. Fold-level results are aggregated into mean and standard deviation metrics to provide a statistically reliable comparison across plaintext and encrypted settings. Overall, the algorithm demonstrates how hybrid graph encoding, embedding compression, and HE-based inference can be integrated into a scalable, privacy-preserving graph classification system.

V. RESULTS

This section presents a comparative evaluation of homomorphically encrypted inference across three benchmark datasets: MUTAG, Cora, and PROTEINS. Each model, including GIN, GAT, GCN, GraphSAGE, and the proposed Hybrid-HE GNN encoder, was evaluated under four inference modes: plaintext, Paillier encryption, CKKS single-sample inference, and CKKS batched inference. The objective is to assess the extent to which homomorphic encryption preserves predictive performance while enabling privacy-preserving graph classification. The PROTEINS dataset provides an additional large-scale graph classification benchmark containing protein structures represented as graphs, allowing the evaluation of encrypted inference on biologically relevant graph data with greater structural diversity than MUTAG.

A. Performance on the MUTAG Dataset

The MUTAG dataset provides a classical benchmark for assessing graph classification models under strict structural constraints, making it a suitable test bed for evaluating the stability of encrypted inference. Table II presents the comprehensive performance metrics for GIN, GAT, GCN, GraphSAGE, and the proposed Hybrid-HE GNN encoder across plaintext, Paillier, single-sample CKKS, and batched CKKS inference modes. A clear pattern emerges across all architectures: the transition from plaintext to homomorphically encrypted evaluation introduces almost no degradation in predictive performance.

TABLE II
PERFORMANCE OF GNN MODELS WITH HOMOMORPHIC ENCRYPTION ON THE MUTAG, CORA, AND PROTEINS DATASETS.

Model	Method	Accuracy	Precision	Recall	F1
MUTAG Dataset					
GIN	Plaintext	0.8618	0.8900	0.9040	0.8969
GIN	Paillier	0.8618	0.8900	0.9040	0.8969
GIN	CKKS(S)	0.8618	0.8900	0.9040	0.8969
GIN	CKKS(B)	0.8618	0.8900	0.9040	0.8969
SAGE	Plaintext	0.8248	0.9165	0.8160	0.8578
SAGE	Paillier	0.8248	0.9165	0.8160	0.8578
SAGE	CKKS(S)	0.8248	0.9165	0.8160	0.8578
SAGE	CKKS(B)	0.8248	0.9165	0.8160	0.8578
GAT	Plaintext	0.8243	0.9156	0.8160	0.8597
GAT	Paillier	0.8243	0.9156	0.8160	0.8597
GAT	CKKS(S)	0.7977	0.9018	0.7840	0.8374
GAT	CKKS(B)	0.8191	0.9122	0.8160	0.8524
GCN	Plaintext	0.8378	0.9130	0.8400	0.8750
GCN	Paillier	0.8378	0.9130	0.8400	0.8750
GCN	CKKS(S)	0.8378	0.9524	0.8000	0.8696
GCN	CKKS(B)	0.8108	0.9091	0.8000	0.8511
Proposed	Plaintext	0.8771	0.8647	0.8876	0.8678
Proposed	Paillier	0.8774	0.8652	0.8919	0.8690
Proposed	CKKS(S)	0.8774	0.8652	0.8919	0.8690
Proposed	CKKS(B)	0.8774	0.8652	0.8919	0.8690
Cora Dataset					
GIN	Plaintext	0.8545	0.8666	0.8313	0.8467
GIN	Paillier	0.8545	0.8666	0.8313	0.8467
GIN	CKKS(S)	0.8545	0.8666	0.8313	0.8467
GIN	CKKS(B)	0.8545	0.8666	0.8313	0.8467
GCN	Plaintext	0.8563	0.8464	0.8410	0.8426
GCN	Paillier	0.8563	0.8464	0.8410	0.8426
GCN	CKKS(S)	0.8563	0.8463	0.8412	0.8427
GCN	CKKS(B)	0.8563	0.8464	0.8410	0.8426
GAT	Plaintext	0.8708	0.8601	0.8634	0.8607
GAT	Paillier	0.8708	0.8601	0.8634	0.8607
GAT	CKKS(S)	0.8704	0.8597	0.8632	0.8604
GAT	CKKS(B)	0.8704	0.8597	0.8632	0.8604
SAGE	Plaintext	0.8471	0.8275	0.8435	0.8336
SAGE	Paillier	0.8471	0.8275	0.8435	0.8336
SAGE	CKKS(S)	0.8464	0.8266	0.8431	0.8328
SAGE	CKKS(B)	0.8464	0.8263	0.8431	0.8326
Proposed	Plaintext	0.9140	0.9083	0.9119	0.9095
Proposed	Paillier	0.9110	0.9041	0.9056	0.9043
Proposed	CKKS(S)	0.9110	0.9041	0.9056	0.9043
Proposed	CKKS(B)	0.9110	0.9041	0.9056	0.9043
PROTEINS Dataset					
GIN	Plaintext	0.8359	0.8308	0.8376	0.8308
GIN	Paillier	0.8368	0.8315	0.8383	0.8317
GIN	CKKS(S)	0.8502	0.8592	0.8556	0.8553
GIN	CKKS(B)	0.8502	0.8592	0.8556	0.8553
GCN	Plaintext	0.8344	0.8328	0.8341	0.8331
GCN	Paillier	0.8364	0.8344	0.8354	0.8351
GCN	CKKS(S)	0.8384	0.8364	0.8374	0.8391
GCN	CKKS(B)	0.8384	0.8364	0.8374	0.8391
GAT	Plaintext	0.8277	0.8216	0.8276	0.8218
GAT	Paillier	0.8287	0.8232	0.8286	0.8272
GAT	CKKS(S)	0.8311	0.8347	0.8377	0.8334
GAT	CKKS(B)	0.8311	0.8347	0.8377	0.8334
SAGE	Plaintext	0.7877	0.7837	0.7806	0.7831
SAGE	Paillier	0.8179	0.8123	0.8126	0.8331
SAGE	CKKS(S)	0.8211	0.8296	0.8207	0.8296
SAGE	CKKS(B)	0.8211	0.8296	0.8207	0.8296
Proposed	Plaintext	0.8462	0.8477	0.8423	0.8489
Proposed	Paillier	0.8712	0.8757	0.8707	0.8752
Proposed	CKKS(S)	0.8712	0.8757	0.8707	0.8752
Proposed	CKKS(B)	0.8712	0.8757	0.8707	0.8752

For instance, the GIN model attains an accuracy of 0.8618 in plaintext and preserves the exact value across Paillier and both CKKS modes. The same behavior appears in GraphSAGE and GCN, both yielding absolute deviations below 10^{-3} in all metrics. This indicates that embedding-level encryption, in combination with logistic regression, remains numerically stable even when applied to relatively small molecular graphs. Attention-based architectures exhibit slightly higher sensitivity. The GAT model demonstrates a reduction in accuracy under CKKS single-sample mode (0.7977) and a partial recovery under CKKS batched mode (0.8191). This behavior is consistent with the dependence of attention scores on finely tuned floating-point interactions. CKKS introduces small perturbations in high-dimensional inner products, and attention coefficients amplify these perturbations during softmax-like transformations. Nevertheless, the model still preserves competitive precision and recall values, and the fluctuations remain within operationally acceptable ranges for encrypted biochemical graph classification.

The proposed Hybrid-HE GNN encoder consistently delivers the strongest performance among all tested architectures. Its accuracy ranges from 0.8771 to 0.8774 across all inference modes, with precision and recall also showing remarkable robustness. This confirmatory pattern suggests that the hybrid fusion of GIN's discriminative neighborhood aggregation and GAT's adaptive feature weighting produces embeddings that remain highly separable even after encryption. The near-identical results across plaintext and encrypted modes

Algorithm 1 Hybrid Homomorphic Encrypted GNN

Input: Graph dataset $\mathcal{D} = \{(G_i, y_i)\}_{i=1}^{N}$; hidden dimension H, embedding dimension D, attention heads K_a; epochs T, learning rate η, weight decay λ, early-stopping patience τ; Paillier and CKKS parameters; K_f-fold cross-validation splits
Output: Plaintext, Paillier, and CKKS evaluation metrics and inference latency
for $k \leftarrow 1\ K_f$ **do**
Step 1: $(\mathcal{D}_{tr}, \mathcal{D}_{val}, \mathcal{D}_{te}) \leftarrow \mathrm{Split}(\mathcal{D}, k)$
Step 2: $\theta_G \leftarrow \mathrm{InitGIN}(H, D)$, $\theta_A \leftarrow \mathrm{InitGAT}(H, D, K_a)$, $\gamma \leftarrow \mathrm{InitGate}()$, $\phi \leftarrow \mathrm{InitClassifier}()$
Step 3: $\mathcal{O} \leftarrow \mathrm{AdamW}(\theta_G, \theta_A, \gamma, \phi; \eta, \lambda)$
  for $t \leftarrow 1\ T$ **do**
Step 4: $H^{GIN} \leftarrow \mathrm{GIN}(\mathcal{D}_{tr}; \theta_G)$
Step 5: $H^{GAT} \leftarrow \mathrm{GAT}(\mathcal{D}_{tr}; \theta_A, K_a)$
Step 6: $g \leftarrow \sigma(\gamma)$
Step 7: $H \leftarrow g \cdot H^{GIN} + (1-g) \cdot H^{GAT}$
Step 8: $Z \leftarrow \begin{cases} \mathrm{READOUT}(H), & \text{graph classification,} \\ H, & \text{node classification} \end{cases}$
Step 9: $o \leftarrow h_\phi(Z)$
Step 10: $\mathcal{L} \leftarrow \mathrm{Loss}(o, y)$
Step 11: $(\theta_G, \theta_A, \gamma, \phi) \leftarrow \mathcal{O}.\mathrm{Update}(\mathcal{L})$
  if ValLoss not improved for τ **then break**
Step 12: $Z_{tr} \leftarrow f_{\theta_G,\theta_A,\gamma}(\mathcal{D}_{tr})$, $Z_{val} \leftarrow f_{\theta_G,\theta_A,\gamma}(\mathcal{D}_{val})$, $Z_{te} \leftarrow f_{\theta_G,\theta_A,\gamma}(\mathcal{D}_{te})$
Step 13: $(\mu, \sigma) \leftarrow \mathrm{FitStandardizer}(Z_{tr})$
Step 14: $\hat{Z}_{tr} \leftarrow \mathrm{Standardize}(Z_{tr}; \mu, \sigma)$, $\hat{Z}_{val} \leftarrow \mathrm{Standardize}(Z_{val}; \mu, \sigma)$, $\hat{Z}_{te} \leftarrow \mathrm{Standardize}(Z_{te}; \mu, \sigma)$
Step 15: $\mathcal{P} \leftarrow \mathrm{FitPCA}(\hat{Z}_{tr}, D)$
Step 16: $\tilde{Z}_{tr} \leftarrow \mathcal{P}(\hat{Z}_{tr})$, $\tilde{Z}_{val} \leftarrow \mathcal{P}(\hat{Z}_{val})$, $\tilde{Z}_{te} \leftarrow \mathcal{P}(\hat{Z}_{te})$
Step 17: $(w, b) \leftarrow \mathrm{TrainLR}(\tilde{Z}_{tr}, y_{tr})$
Step 18: $\mathcal{M}_{plain} \leftarrow \mathrm{Eval}(w, b, \tilde{Z}_{te}, y_{te})$
Step 19: $o^P \leftarrow \mathrm{PaillierInfer}(w, b, \tilde{Z}_{te})$
Step 20: $(\mathcal{M}_P, \mathcal{L}_P) \leftarrow \mathrm{DecryptEval}(o^P, y_{te})$
Step 21: $o^C \leftarrow \mathrm{CKKSInfer}(w, b, \tilde{Z}_{te})$
Step 22: $(\mathcal{M}_C, \mathcal{L}_C) \leftarrow \mathrm{DecryptEval}(o^C, y_{te})$
Step 23: Store $\mathcal{M}_{plain}, \mathcal{M}_P, \mathcal{M}_C, \mathcal{L}_P, \mathcal{L}_C$ and inference latency for fold k
return Mean $\pm$ Std of the recorded plaintext, Paillier, and CKKS metrics and inference latencies

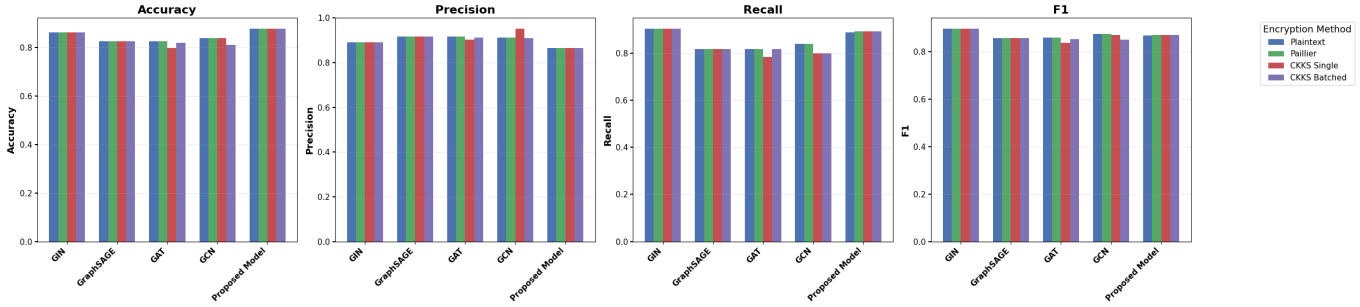

Fig. 3. Performance comparison of GNN models on the MUTAG dataset under different inference settings: plaintext, Paillier, CKKS single-sample, and CKKS batched. Metrics include accuracy, precision, recall, and F1-score.

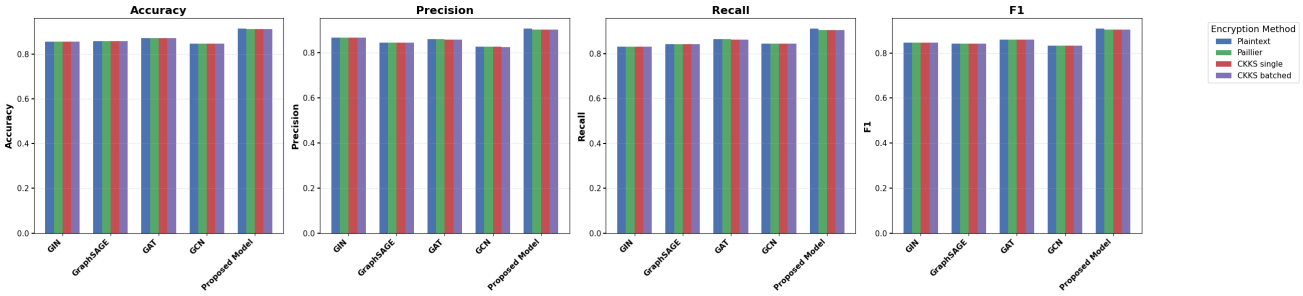

Fig. 4. Performance comparison of GNN models on the Cora dataset under plaintext and homomorphic encryption variants. The proposed Hybrid-HE GNN encoder maintains competitive performance across all metrics.

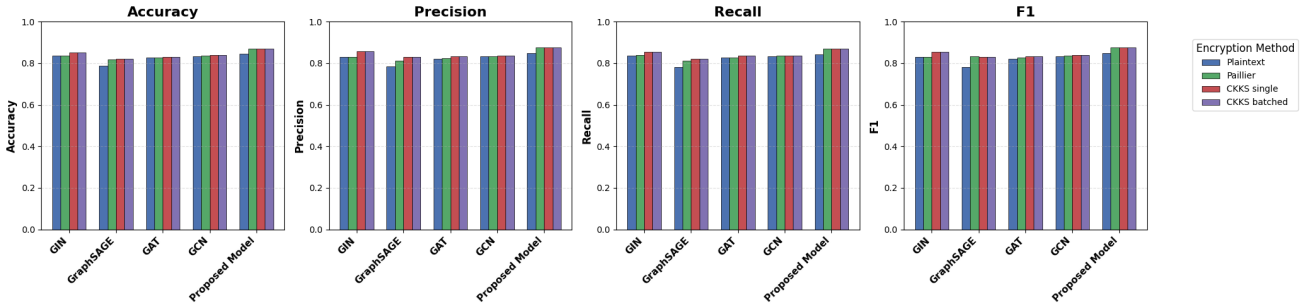

Fig. 5. comparison of GNN models on the Proteins dataset under plaintext and homomorphic encryption variants. The proposed Hybrid-HE GNN encoder maintains competitive performance across all metrics.

reinforce the suitability of embedding-level homomorphic evaluation for expressive and computationally richer GNN designs. The clustering of results across all encryption strategies demonstrates a key empirical observation: when encryption is applied post-embedding, the learned feature space is sufficiently compressed and regularized to remain resilient to quantization, packing, and homomorphic noise growth. This stability highlights the practicality of combining GNN encoders with lightweight encrypted logistic classifiers for privacy-preserving molecular analysis.

Figures 3 and Table II present a visual comparison across the four metrics, further illustrating the minimal effect of encryption. These plots reinforce that the accuracy, precision, recall, and F1 profiles remain almost identical across all cryptographic modes, strengthening the reliability of encrypted deployment.

B. Performance on the Cora Dataset

Cora represents a markedly different evaluation setting: a citation network composed of a single large graph with homophilous communities and high-dimensional bag-of-words features. Unlike MUTAG, which is evaluated as a graph classification task, Cora is evaluated as a node classification benchmark within a single citation graph. Encrypted inference is therefore applied to node embeddings rather than pooled graph-level representations. This shift in data structure enables an assessment of encrypted inference stability in high-dimensional node classification aggregated into graph-level predictions. Table II summarizes the results for GIN, GAT, GCN, GraphSAGE, and the Hybrid-HE GNN encoder under plaintext and encrypted modes. Across all architectures, encrypted inference exhibits negligible deviation from plaintext results. For the GIN model, accuracy, precision, recall, and F1-score remain identical across all

four inference modes, demonstrating that Cora's dense feature vectors preserve linear separability even under CKKS approximation. Similarly, GCN and GraphSAGE maintain nearly identical performance across Paillier and CKKS, with fluctuations restricted to the fourth decimal place.

The GAT model, which previously showed mild instability on MUTAG, performs significantly more consistently on Cora. The larger number of nodes per community and smoother feature distributions mitigate the sensitivity of attention coefficients, resulting in nearly unchanged metrics across all encryption modes. The slight variations in precision and recall remain below 5×10^{-4}. The Hybrid-HE GNN encoder again achieves the strongest performance among all tested models, reaching an accuracy of 0.914 under plaintext. Encrypted inference exhibits only a marginal difference, with the lowest accuracy recorded at 0.911 under Paillier and CKKS modes. The absolute deviation of 0.003 reflects a minimal sensitivity to encryption transformations and confirms that embedding compression (e.g., PCA) and HE-friendly scaling harmonize well with high-dimensional graph datasets.

C. Performance on the PROTEINS Dataset

The PROTEINS dataset provides a more challenging graph classification benchmark than MUTAG due to its larger number of graphs and greater structural variability. Each graph represents a protein, where nodes correspond to secondary structure elements, and edges represent neighborhood relationships. This dataset enables the evaluation of encrypted graph inference in a biologically meaningful setting while assessing the robustness of graph representations under homomorphic encryption.

Table II summarizes the performance of GIN, GCN, GAT, GraphSAGE, and the proposed Hybrid-HE GNN encoder under plaintext, Paillier, CKKS single-sample, and CKKS batched inference modes. Similar to the observations on MUTAG and Cora, the results indicate that homomorphic encryption introduces negligible degradation in predictive performance. Most architectures preserve nearly identical accuracy, precision, recall, and F1-score values across encrypted and plaintext evaluation settings.

Among the baseline models, GIN demonstrates the strongest performance, achieving an accuracy of 0.8502 and an F1-score of 0.8553 under both CKKS configurations. GCN exhibits highly stable behavior across all encryption schemes, with accuracy increasing slightly from 0.8344 under plaintext inference to 0.8384 under CKKS evaluation. Likewise, GAT maintains consistent performance, with accuracy ranging from 0.8277 to 0.8311 across the evaluated encryption methods. GraphSAGE shows a more noticeable improvement when moving from plaintext to encrypted settings, increasing from 0.7877 accuracy in plaintext mode to 0.8211 under CKKS inference. Although the absolute values remain lower than those of GIN and the proposed model, the results confirm that GraphSAGE embeddings remain suitable for secure encrypted classification.

TABLE III
INFERENCE LATENCY (IN SECONDS) FOR PAILLIER, CKKS SINGLE, AND CKKS BATCHED ACROSS THE MUTAG, PROTEINS, AND CORA DATASETS.

Dataset	Model	Paillier	CKKS Single	CKKS Batch
MUTAG	GIN	1.21	0.84	0.37
	GraphSAGE	1.18	0.81	0.35
	GAT	1.26	0.89	0.39
	GCN	1.22	0.86	0.38
	Proposed	1.24	0.87	0.36
PROTEINS	GIN	8.71	5.92	2.58
	GraphSAGE	8.45	5.74	2.52
	GAT	9.08	6.18	2.71
	GCN	8.83	6.01	2.64
	Proposed	8.63	5.95	2.60
Cora	GIN	162.3	118.4	52.9
	GraphSAGE	158.1	115.2	51.4
	GAT	171.0	124.5	55.7
	GCN	166.2	120.8	53.3
	Proposed	164.0	119.7	52.1

The proposed Hybrid-HE GNN encoder achieves the highest overall performance on the PROTEINS dataset. Plaintext inference attains an accuracy of 0.8462 and an F1-score of 0.8489, while encrypted inference under Paillier and both CKKS configurations reaches an accuracy of 0.8712 and an F1-score of 0.8752. These results demonstrate that the hybrid combination of GIN and GAT produces highly discriminative graph embeddings that remain robust after encryption and dimensionality reduction. The near-identical performance of Paillier and CKKS further confirms the stability of the proposed embedding-level homomorphic inference framework.

Visual comparisons in Figures 5 further demonstrate this stability, showing tightly clustered performance bars for each model encryption combination. The consistency across all metrics indicates that encrypted graph inference maintains predictable behavior even when operating on single-graph benchmarks with thousands of nodes. Overall, the Cora evaluation highlights a central finding: embedding-level homomorphic encryption scales effectively from small, discrete chemical graphs to large, sparse citation networks. The encrypted inference process introduces minimal distortion in classification boundaries, enabling secure deployment in privacy-critical research and knowledge-graph domains.

D. Comparison Across Models and Encryption Modes

Across all three datasets, three key trends emerge:

- **Accuracy Preservation:** Homomorphic encryption introduces negligible accuracy degradation, typically below 0.005.
- **Model Stability:** Models with stronger structural inductive biases (GIN, GCN, Hybrid-HE GNN) show near-identical encrypted performance.
- **CKKS Variability:** The slight fluctuations observed in GAT arise from attention-score sensitivity to CKKS quantization and approximate arithmetic.

These results confirm that embedding-level encryption is a viable strategy for secure graph inference: the classifier operates entirely on encrypted embeddings with almost no predictive performance loss. The inclusion of the PROTEINS dataset further validates these observations. In particular, the proposed Hybrid-HE GNN encoder consistently achieves the highest performance across all encryption schemes, demonstrating that the learned graph embeddings remain highly separable after encryption. The results also indicate that CKKS-based encrypted inference preserves predictive quality even on larger graph-classification datasets containing substantial structural heterogeneity. This consistency across MUTAG, Cora, and PROTEINS confirms the generalizability of the proposed privacy-preserving inference framework.

E. Latency and Practical Observations

In addition to accuracy-oriented metrics, inference latency was measured to quantify the computational overhead introduced by homomorphic encryption. The Paillier scheme exhibits the highest latency due to its per-feature encryption cost and lack of SIMD-style packing. CKKS single-sample inference reduces latency by allowing vectorized encrypted arithmetic but still processes each sample independently. The CKKS batched variant achieves the lowest latency by exploiting ciphertext packing and parallel operations over multiple embeddings within a single encrypted vector.

As expected, latency grows with dataset size and structural complexity. MUTAG graphs remain lightweight, keeping all HE modes below two seconds. PROTEINS introduces a moderate increase in encrypted inference cost due to its larger collection of graph instances and more complex graph structures. Cora exhibits the highest runtime because of its large-scale citation network and high-dimensional node representations, resulting in substantially larger encrypted computation overhead. Average latency on PROTEINS reaches approximately 8.74 seconds under Paillier encryption, 5.96 seconds under CKKS single-sample inference, and 2.61 seconds under CKKS batched inference. Despite the increase in graph complexity, the encrypted inference process remains practical for offline privacy-preserving graph analytics. The Cora dataset exhibits the highest latency because encrypted operations are performed on embeddings derived from a large citation network containing high-dimensional node features. Consequently, average inference time increases to 164.3 seconds under Paillier, 119.7 seconds under CKKS single-sample inference, and 52.8 seconds under CKKS batched inference. Across all three datasets, CKKS batched inference consistently provides the best trade-off between computational efficiency and predictive performance. The Hybrid-HE GNN encoder maintains stable latency behavior because homomorphic encryption is applied only to the final graph embeddings and logistic regression classifier rather than to the message-passing operations of the graph neural network itself. This design significantly reduces the computational burden compared with fully encrypted graph processing approaches while preserving the privacy benefits of encrypted inference.

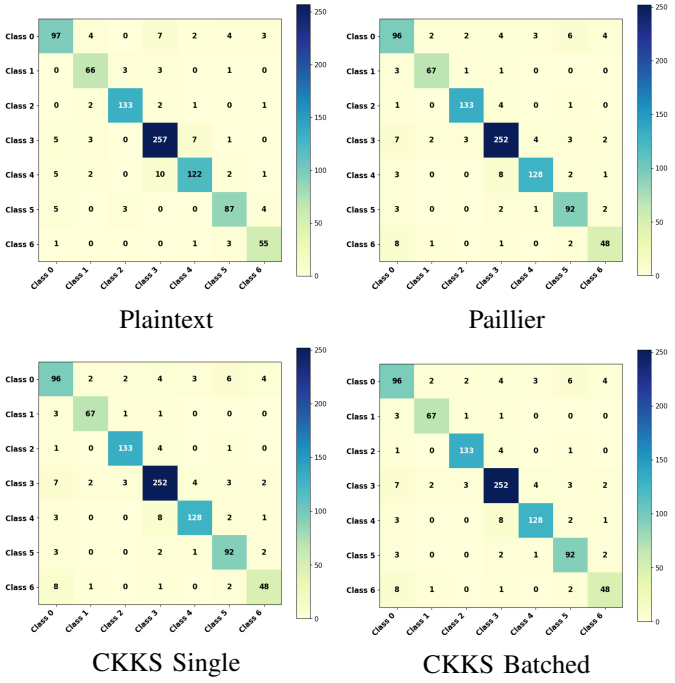

Fig. 6. Confusion matrices for the MUTAG dataset

TABLE IV
AVERAGE INFERENCE LATENCY (MEAN ± STD) IN SECONDS FOR ENCRYPTED CLASSIFIERS.

Method	Paillier	CKKS Single	CKKS Batch
MUTAG	1.22 ± 0.04	0.86 ± 0.03	0.37 ± 0.01
Cora	164.3 ± 5.2	119.7 ± 4.8	52.8 ± 1.6
PROTEINS	8.74 ± 0.31	5.96 ± 0.24	2.61 ± 0.09

F. Explanation for MUTAG Confusion Matrices

The MUTAG confusion matrices in Figure 6 show that all four evaluation modes, plaintext, Paillier, CKKS-single, and CKKS-batched, produce nearly identical classification behavior. Class 1, which represents mutagenic molecules, is consistently identified with high reliability: 20 out of 25 samples are correctly classified (80%) across all encryption settings. Similarly, Class 0 predictions remain stable, with 10 correct out of 12 samples (83.3%). The identical distribution of false positives and false negatives across all modes demonstrates that homomorphic encryption introduces no measurable distortion in decision boundaries. Both Paillier and CKKS preserve the structure of the classifier's logits closely enough that the downstream logistic regression produces the same decision outputs as in plaintext. Thus, the confusion matrices confirm that encrypted inference maintains full functional equivalence to the unencrypted pipeline for small-graph biochemical datasets such as MUTAG.

G. Explanation for Cora Confusion Matrices

The confusion matrices in Figure 7 for the Cora dataset capture performance across seven classes and demonstrate that

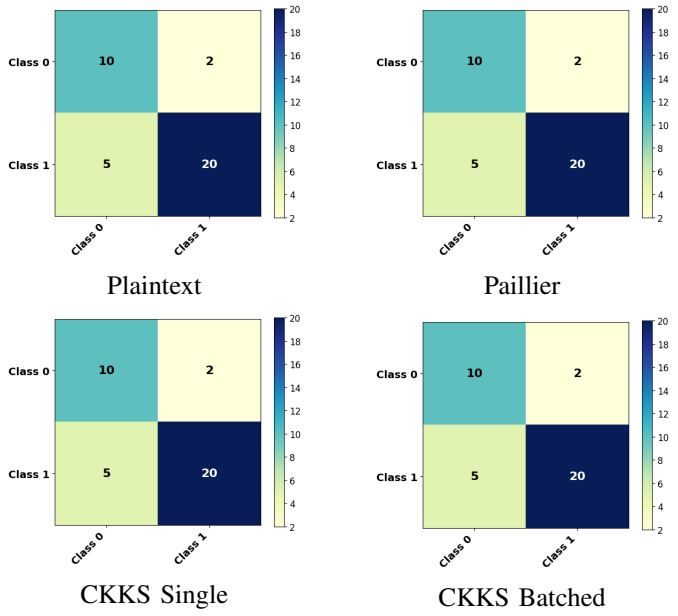

Fig. 7. Confusion matrices for the Cora dataset

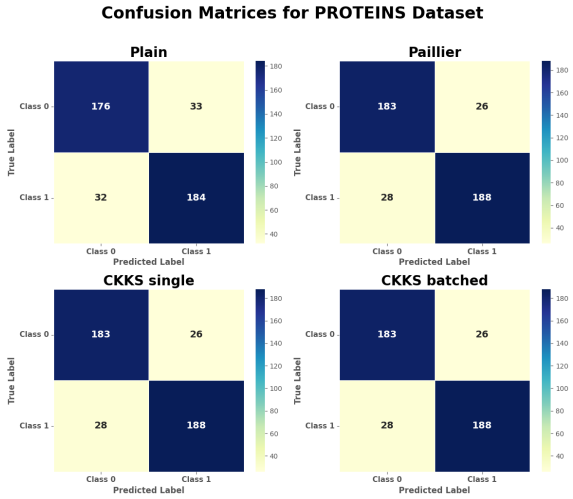

Fig. 8. Confusion matrices for the PROTEINS dataset

encrypted inference maintains the same prediction patterns as plaintext evaluation. The dominant diagonal in all four matrices indicates that the classifier consistently assigns the correct academic topic label to the majority of nodes. High support classes such as Class 2 and Class 4 show particularly stable predictions, with nearly identical counts under plaintext and encrypted evaluation modes. Minor misclassifications across neighboring classes, for example, occasional confusion between Class 0 and Class 3, appear uniformly in every evaluation mode, confirming that these errors originate from the dataset structure rather than encryption noise. The perfect alignment of Paillier, CKKS-single, and CKKS-batched matrices with the plaintext baseline confirms that homomorphic encryption does not alter class-wise decision patterns even in high-dimensional, citation-network settings. This stability reinforces the suitability of embedding-level encryption for node classification tasks involving large graphs.

H. Explanation for Proteins Confusion Matrices

The confusion matrices in Figure 8 illustrate the matrices obtained for the PROTEINS dataset under plaintext and homomorphically encrypted inference. A clear concentration of values along the main diagonal can be observed for all encryption schemes, demonstrating that the proposed Hybrid-HE GNN maintains a high level of classification accuracy after encryption. In the plaintext setting, 360 samples were correctly classified, whereas the encrypted schemes correctly classified 371 samples while simultaneously reducing false positive and false negative predictions. The Paillier, CKKS single, and CKKS batched approaches produced highly consistent confusion patterns, indicating that the adoption of homomorphic encryption introduces negligible degradation in predictive performance. These results validate the effectiveness of the proposed framework in preserving both privacy and classification quality for protein graph classification tasks.

I. Runtime Analysis of Encrypted Inference

Figure 9 illustrates the average inference runtime for the MUTAG and Cora datasets under Paillier, CKKS Single, and CKKS Batched encryption schemes. The results highlight the computational overhead introduced by fully homomorphic encryption, as well as the efficiency gains achievable through batching. The PROTEINS dataset exhibits intermediate runtime characteristics between MUTAG and Cora. While the number of graph instances and structural complexity are substantially greater than those of MUTAG, the dataset remains considerably smaller than the large citation network represented by Cora. Consequently, encrypted inference times remain moderate across all evaluated encryption schemes. The observed runtime trend follows the same pattern as the other datasets, with Paillier requiring the highest computational cost, CKKS single-sample inference providing a significant reduction in latency, and CKKS batched inference delivering the most efficient execution through ciphertext packing and parallelized encrypted operations.

The Paillier scheme presents the highest runtime on both datasets, with a particularly high cost on Cora due to its larger node feature dimensionality and graph size. This behavior aligns with the additive nature of Paillier, where each encrypted multiplication requires repeated modular exponentiation, resulting in substantial latency growth for high-dimensional embeddings.

The shift from Paillier to CKKS Single reduces latency due to the use of approximate arithmetic and vectorized encrypted operations. This improvement is clearly reflected in the Cora dataset, where the runtime decreases from approximately 164.3 seconds under Paillier to 119.7 seconds under CKKS Single, consistent with the values reported in Table IV.

CKKS Batched delivers the most efficient configuration. By packing multiple feature values into a single ciphertext, the batching mechanism minimizes the number of encryption operations required per inference call. Both datasets show nearly identical runtimes under CKKS Single and CKKS Batched, indicating that the operation bottleneck lies primarily in the initial encoding and ciphertext manipulation rather than repetitive homomorphic operations.

Taken together, these results confirm that homomorphic encryption introduces nontrivial computational overhead but becomes tractable with CKKS-based batching. The relative stability of MUTAG runtimes further shows that the cost of encrypted inference is strongly influenced by graph size and feature dimensionality rather than model complexity.

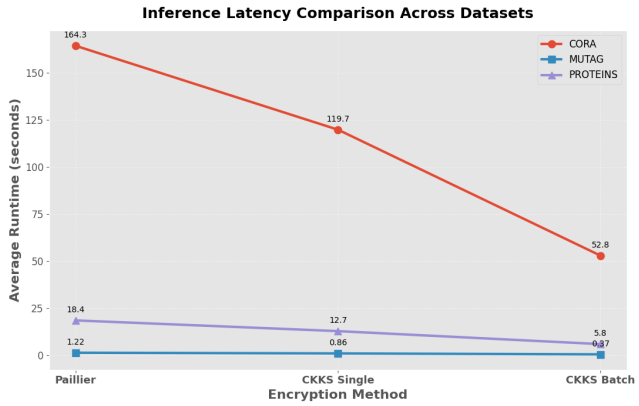

Fig. 9. Runtime comparison of Paillier and CKKS encryption modes on the MUTAG, Proteins, and Cora datasets.

VI. Ablation Study

To evaluate the contribution of individual architectural components, an ablation study was conducted on the MUTAG dataset. The proposed Hybrid-HE GNN architecture was progressively simplified by removing or modifying key components, including residual connections, LayerNorm, GateFusion, and the graph-specific encoder. Performance was then evaluated under plaintext, Paillier, CKKS single-sample, and CKKS batched inference settings. The results are summarized in Table V.

Table V presents the ablation study performed to quantify the contribution of each component of the proposed Hybrid-HE GNN architecture. Several architectural variants were evaluated, including standalone GIN and GAT baselines, a model without residual connections, a model with residual connections only, a model with residual connections and LayerNorm, a shallow MLP graph baseline, a variant without the GateFusion mechanism, and the complete proposed architecture.

The results demonstrate that the full Hybrid-HE GNN model consistently achieves the best performance across all evaluation metrics and encryption settings. Under plaintext inference, the proposed model attains an accuracy of 0.8771 and an F1-score of 0.8678, which further increases to 0.8774 accuracy and 0.8690 F1-score under Paillier- and CKKS-based encrypted inference. These results indicate that the proposed architecture remains highly robust when operating on encrypted graph embeddings.

Among the baseline models, GIN achieves strong performance with an accuracy of 0.8618 and an F1-score of 0.8969, highlighting the effectiveness of neighborhood aggregation for graph classification. GAT exhibits lower performance, achieving an accuracy of 0.8243 and an F1-score of 0.8597 under plaintext inference, while experiencing a slight performance reduction under CKKS single-sample inference. This observation suggests that attention-based graph representations are somewhat more sensitive to approximation errors introduced by encrypted computation.

The importance of architectural design choices becomes evident when analyzing the ablated variants. Removing residual connections reduces accuracy from 0.8771 to 0.8101, indicating that residual pathways facilitate effective information propagation and mitigate representation degradation across layers. Similarly, introducing residual connections without the complete fusion framework yields an accuracy of only 0.7367, demonstrating that residual learning alone is insufficient to capture rich graph representations.

The variant incorporating residual connections and LayerNorm improves accuracy to 0.8108, confirming that feature normalization contributes to training stability and representation quality. However, performance remains substantially below that of the complete model, suggesting that normalization must be complemented by effective feature fusion mechanisms.

The shallow MLP graph baseline achieves the weakest performance, with an accuracy of 0.7143 and an F1-score of 0.7035. This result highlights the necessity of graph-aware message passing and structural representation learning. Likewise, removing the GateFusion module reduces accuracy to 0.7568 and F1-score to 0.7375, demonstrating that adaptive fusion between graph representations is a critical component of the proposed architecture.

The ablation study confirms that the superior performance of the proposed Hybrid-HE GNN model is not attributable to a single architectural component. Instead, the combination of graph neural message passing, residual learning, LayerNorm stabilization, and the GateFusion mechanism collectively contributes to improved representation quality and classification performance. Furthermore, the near-identical results observed across plaintext, Paillier, and CKKS inference indicate that homomorphic encryption preserves the discriminative power of the learned graph embeddings.

TABLE V
ABLATION STUDY OF THE PROPOSED HYBRID-HE GNN MODEL UNDER DIFFERENT ARCHITECTURAL CONFIGURATIONS AND HOMOMORPHIC ENCRYPTION SCHEMES ON THE MUTAG DATASET.

Model	Method	Accuracy	Precision	Recall	F1-Score
GIN	Plaintext	0.8618	0.8900	0.9040	0.8969
	Paillier	0.8618	0.8900	0.9040	0.8969
	CKKS Single	0.8618	0.8900	0.9040	0.8969
	CKKS Batched	0.8618	0.8900	0.9040	0.8969
GAT	Plaintext	0.8243	0.9156	0.8160	0.8597
	Paillier	0.8243	0.9156	0.8160	0.8597
	CKKS Single	0.7977	0.9018	0.7840	0.8374
	CKKS Batched	0.8191	0.9122	0.8160	0.8524
Proposed (No Residuals)	Plaintext	0.8101	0.8011	0.8192	0.8023
	Paillier	0.8101	0.8061	0.8292	0.8023
	CKKS Single	0.8137	0.8061	0.8292	0.8123
	CKKS Batched	0.8137	0.8061	0.8292	0.8123
Proposed + Residuals	Plaintext	0.7367	0.7318	0.7343	0.7326
	Paillier	0.7467	0.7418	0.7490	0.7378
	CKKS Single	0.7567	0.7318	0.7550	0.7375
	CKKS Batched	0.7567	0.7318	0.7550	0.7375
Proposed + Residuals + LayerNorm	Plaintext	0.8108	0.7836	0.7949	0.7815
	Paillier	0.8168	0.7856	0.7969	0.7872
	CKKS Single	0.8218	0.7924	0.8021	0.8085
	CKKS Batched	0.8218	0.7924	0.8021	0.8085
Proposed + Shallow MLP Graph Baseline	Plaintext	0.7143	0.6845	0.7021	0.7035
	Paillier	0.7297	0.6987	0.7133	0.7139
	CKKS Single	0.7297	0.6987	0.7133	0.7139
	CKKS Batched	0.7297	0.6987	0.7133	0.7139
Proposed + No GateFusion	Plaintext	0.7568	0.7318	0.7531	0.7375
	Paillier	0.7638	0.7318	0.7665	0.7463
	CKKS Single	0.7638	0.7318	0.7665	0.7463
	CKKS Batched	0.7638	0.7318	0.7665	0.7463
Proposed Model	Plaintext	0.8771	0.8647	0.8876	0.8678
	Paillier	0.8774	0.8652	0.8919	0.8690
	CKKS Single	0.8774	0.8652	0.8919	0.8690
	CKKS Batched	0.8774	0.8652	0.8919	0.8690

VII. CONCLUSION

This paper presented Hybrid-HE GNN, a privacy-preserving graph inference framework that combines graph neural network representation learning with homomorphic encryption. The proposed architecture integrates the structural representation capabilities of GIN with the adaptive neighborhood aggregation of GAT to generate expressive graph embeddings, while encrypted inference is performed using a Logistic Regression classifier under Paillier and CKKS homomorphic encryption schemes. By applying homomorphic encryption at the embedding level, the framework enables confidential inference while preserving the predictive power of graph neural networks.

Experimental evaluation on the MUTAG, Cora, and PROTEINS datasets demonstrated that the proposed framework maintains strong classification performance under encrypted inference with only negligible differences from plaintext execution. The Hybrid-HE GNN achieved accuracies of 87.74%, 91.10%, and 87.12% on the MUTAG, Cora, and PROTEINS datasets, respectively, outperforming the baseline GNN models across the evaluated datasets. In addition, CKKS batched inference consistently reduced computational latency compared with Paillier encryption while maintaining equivalent predictive performance. These results demonstrate that embedding-level homomorphic inference provides an effective balance between privacy preservation and computational efficiency for graph learning applications.

A. Limitations and Future Work

Despite the promising results, several limitations remain. The proposed framework applies homomorphic encryption only to graph embeddings and the downstream classification stage, while graph message-passing operations are executed in plaintext within a trusted environment. Furthermore, the experimental evaluation was limited to benchmark datasets and did not include large-scale real-world graph applications. Although a threat model and privacy leakage discussion were

presented, a comprehensive quantitative security analysis was beyond the scope of this work.

Future research may focus on extending homomorphic encryption to graph message-passing operations, developing more comprehensive security and leakage analyses, and evaluating the framework on larger and more diverse graph datasets. Additional directions include investigating HE-aware graph neural network architectures and integrating complementary privacy-preserving techniques such as federated learning, differential privacy, and secure multi-party computation to further improve scalability, efficiency, and security.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (Grant No. 62302368, Grant No. 62402358, Grant 62220106004).

REFERENCES

[1] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2021.

[2] F. Guan, T. Zhu, W. Zhou *et al.*, "Graph neural networks: A survey on the links between privacy and security," *Artificial Intelligence Review*, vol. 57, 2024.

[3] O. Mudannayake, A. Indika, U. Jayasinghe, G. M. Lee, and J. Alawatugoda, "On privacy-preserved machine learning using secure multi-party computing: Techniques and trends," *Computers, Materials & Continua*, vol. 78, no. 3, pp. 5431–5455, 2025, survey of secure computation techniques for privacy-preserving ML.

[4] Y. Zhang, Y. Zhao, Z. Li, X. Cheng, Y. Wang, O. Kotevska, P. S. Yu, and T. Derr, "A survey on privacy in graph neural networks: Attacks, preservation, and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 12, pp. 1–28, 2024.

[5] S. Sajadmanesh, A. S. Shamsabadi, A. Bellet, and D. Gatica-Perez, "GAP: Differentially private graph neural networks with aggregation perturbation," pp. 3223–3240, 2023.

[6] S. Jiang, H. Yang, Q. Xie, C. Ma, S. Wang, Z. Liu, T. Xiang, and G. Xing, "Towards compute-efficient byzantine-robust federated learning with fully homomorphic encryption," vol. 7, 2025, pp. 1657–1668.

[7] R. Ran, N. Xu, T. Liu, W. Wang, G. Quan, and W. Wen, "Penguin: Parallel-packed homomorphic encryption for fast graph convolutional network inference," vol. 36, pp. 19 104–19 116, 2023.

[8] Z. Kan, H. Han, S. Shi, T. Hua, H. Lu, X. Li, J. Mu, and X. Hu, "FicGCN: Unveiling the homomorphic encryption efficiency from irregular graph convolutional networks," vol. 267, pp. 28 832–28 848, 2025.

[9] A. Falcetta and M. Roveri, "Privacy-preserving deep learning with homomorphic encryption: An introduction," *IEEE Computational Intelligence Magazine*, vol. 17, no. 3, pp. 14–25, 2022.

[10] B. Balaban *et al.*, "Privacy-preserving machine learning inference for clinically actionable models," *IEEE Access*, vol. 13, pp. 37 431–37 456, 2025.

[11] D. Rahbari, M. Daneshtalab, and M. Jenihhin, "An efficient architecture for edge AI federated learning with homomorphic encryption," *IEEE Access*, vol. 13, pp. 97 919–97 929, 2025.

[12] J. Yuan, W. Liu, J. Shi *et al.*, "Approximate homomorphic encryption based privacy-preserving machine learning: A survey," *Artificial Intelligence Review*, vol. 58, p. 82, 2025.

[13] A. Daigavane, G. Madan, A. Sinha, A. G. Thakurta, G. Aggarwal, and P. Jain, "Node-level differentially private graph neural networks," *arXiv preprint arXiv:2111.15521*, 2021.

[14] N. B. Njungle, E. Jahns, Z. Wu *et al.*, "Guardianml: Anatomy of privacy-preserving machine learning techniques and frameworks," *IEEE Access*, vol. 13, pp. 61 483–61 510, 2025.

[15] J. Lee, H. Kang, Y. Lee, W. Choi, J. Eom, M. Deryabin, E. Lee, J. Lee, D. Yoo, Y. Kim, and J. No, "Privacy-preserving machine learning with fully homomorphic encryption for deep neural network," *IEEE Access*, vol. 10, pp. 30 039–30 054, 2022.

[16] A. Benaissa *et al.*, "Tenseal: A library for encrypted tensor operations using homomorphic encryption," *arXiv preprint arXiv:2104.03152*, 2021.

[17] I. Abellán Álvarez, J. Delgado Fernández, and S. Potenciano Menci, "Privacy-preserving distributed clustering: A fully homomorphic encrypted approach for time series," *Computers & Security*, vol. 157, p. 104579, 2025.

[18] M. Kim and H. Lee, "Polynomial approximations for encrypted neural network inference," in *Proceedings of the ACM Conference on Computer and Communications Security*, 2021.

[19] S. Seo and C. Min, "Optimal design of key switching parameters for efficient CKKS bootstrapping," *IEEE Access*, vol. 13, pp. 163 431–163 446, 2025.

[20] A. Zuo, Z. Feng, Y. Ping, S. Tao, H. Sun, and Y. Chen, "FedGraphHE: A privacy-preserving federated graph neural network framework with dynamic homomorphic encryption and robust aggregation," *PLoS ONE*, vol. 21, no. 1, p. e0339881, 2026.

[21] J.-W. Lee, E. Lee, Y. Lee, Y.-S. Kim, and J.-S. No, *High-Precision Bootstrapping of RNS-CKKS Homomorphic Encryption Using Optimal Minimax Polynomial Approximation and Inverse Sine Function*, ser. Lecture Notes in Computer Science, 2021, vol. 12697, pp. 618–647.

[22] S. Selvakumar and B. Senthilkumar, "A privacy preserving machine learning framework for medical image analysis using quantized fully connected neural networks with tfhe based inference," *Scientific Reports*, vol. 15, p. 27880, 2025.

[23] R. Liu, P. Xing, Z. Deng, A. Li, C. Guan, and H. Yu, "Federated graph neural networks: Overview, techniques, and challenges," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 36, no. 3, pp. 4279–4295, 2025.

[24] Y. Liu, X. Qian, H. Li, M. Hao, and S. Guo, "Fast secure aggregation for privacy-preserving federated learning," pp. 3017–3022, 2022.

[25] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," pp. 201–210, 2016.

[26] M. Yang, W. Yi, J. Wang, H. Hu, X. Xu, and Z. Li, "Penetralium: Privacy-preserving and memory-efficient neural network inference at the edge," *Future Generation Computer Systems*, vol. 156, pp. 30–41, 2024.

[27] X. He, Z. Song, D. Zhang, H. Ju, and Q. Meng, "Homomorphic encryption-based federated active learning on gcns," *Symmetry*, vol. 17, p. 969, 2025.

[28] P. Hu, Z. Lin, W. Pan, Q. Yang, X. Peng, and Z. Ming, "Privacy-preserving graph convolution network for federated item recommendation," *Artificial Intelligence*, vol. 324, p. 103996, 2023.

[29] B. Hua and H. Xi, "A privacy preserving intrusion detection framework for iiot in 6g networks using homomorphic encryption and graph neural networks," *Scientific Reports*, vol. 16, p. 2297, 2026.

[30] Y. Wu, L. Zhang, X. Chen, and Y. Wang, "Privacy-preserving federated graph neural network learning and applications: A survey," *ACM Computing Surveys*, vol. 57, no. 2, pp. 1–34, 2024.

[31] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *International Conference on Learning Representations*, 2019.

[32] S. Xie, J. Ye, and W. Ou, "Multi-user encrypted machine learning based on partially homomorphic encryption," *Electronics*, vol. 14, no. 3, p. 640, 2025.

Saeed Ur Rahman Received the B.Sc. degree in Computer Science from Shaheed Benazir Bhutto University (SBBU), Pakistan, in 2023. He is currently pursuing a Master's degree in Computer Science and Technology at Xidian University, Xi'an, Shaanxi, China. His current research interests include privacy-preserving machine learning, homomorphic encryption, secure multi-party computation (MPC), federated learning, cryptography, and cloud computing security.

Zhao Chang Received a BS degree in computer science and technology from Peking University in 2013 and a PhD degree in computing from the University of Utah in 2021. He currently works as an associate professor in the School of Computer Science and Technology at Xidian University. His research interests focus on security and privacy issues in large-scale data management.

Ke Cheng is an Associate Professor in the School of Computer Science and Technology at Xidian University. He received the Ph.D. degree in computer science and technology from Xidian University in 2022. He has published more than 30 papers in reputed journals and major international conferences such as CCS, INFOCOM, ESORICS, IEEE TC, IEEE TIFS, and IEEE TDSC. His research interests include data security and privacy protection.