



QGCNLP: Hybrid Quantum–classical Graph Convolutional Network based Link Prediction[☆]

Nisha Singh^a , Mukesh Kumar^b , Jaideep Sharma^a , Bhaskar Biswas^a

^a Department of Computer Science and Engineering, Indian Institute of Technology (BHU), Varanasi, India

^b School of Computer Engineering, Kalinga Institute of Industrial Technology (KIIT), Deemed to be University, Bhubaneswar, Odisha, India

ARTICLE INFO

Dataset link: [Pytorch Geometric Dataset \(Reference data\)](#)

Keywords:

Link prediction
Social networks
Deep learning
Graph Convolutional Network (GCN)
Quantum computing

ABSTRACT

Recently, graph convolutional networks (GCN) have become very popular due to the fact that they are an effective way to perform convolution on a non-Euclidean space-like graph and that convolution operations on neural networks have produced remarkable results. However, GCN experiences speed and memory limitations on traditional computing platforms as the network's size and complexity increase. Quantum computing, on the other hand, has proven to be very effective in increasing expressiveness by exploring the high-dimensional Hilbert space for complex correlation, alongside providing extraordinarily high computational parallelism. As a result, there is considerable potential for integrating these two advanced technologies. Therefore, we suggested a novel hybrid Quantum–classical Graph Convolutional network-based Link Prediction (QGCNLP) model for the complex task of Link Prediction. This model applies quantum enhancements in a very unique way; instead of converting all input features to quantum data, it applies quantum circuits to the aggregated features obtained after graph convolution in every GCN layer. The quantum-enhanced aggregated values are passed repeatedly till the readout layer, just like the traditional GCN message passing, and then optimized for minimizing the link prediction loss. This approach boosts the model's performance without demanding substantial quantum resources, making it ideal for the present NISQ era. We firmly demonstrate that our suggested methods are superior after doing rigorous testing on several static datasets as well as dynamic ones with a number of performance evaluation metrics over varied learning rates, qubit counts, circuit variations, a number of benchmark classical models, and state-of-the-art methods.

1. Introduction

The emerging influence of online social networks is evident these days, and it provides researchers with an opportunity to develop solutions to several practical problems. The graph-based learning tasks in Network Science, such as node identification and categorization, link prediction, community detection, graph categorization, graph summarization, and many more, were effectively resolved by a number of machine learning and hybrid models [1]. Numerous real-world uses for these works included spotting well-known influencers or spammers on social media [2], suggesting products on e-commerce platforms [3], ranking web pages in search results [4], forecasting call durations [5] and summarizing traffic or usage trends for anomaly detection [6] in

[☆] This article is part of a Special issue entitled: 'LINK PREDICTION' published in Physica A.

* Corresponding author.

E-mail addresses: nishasingh.rs.cse21@itbhu.ac.in (N. Singh), mukesh.kumarfcs@kiit.ac.in (M. Kumar), jaideep.sharma.cse20@iitbhu.ac.in (J. Sharma), bhaskar.cse@itbhu.ac.in (B. Biswas).

<https://doi.org/10.1016/j.physa.2026.131449>

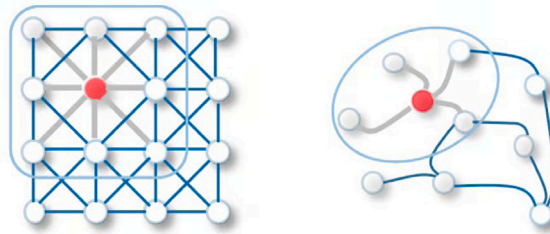


Fig. 1. Convolution in CNN Vs GCN visualization introduced by Wu et al. (2020) [25].

communication networks, IoT device management and service recommendation [7], topic detection with graph classification in knowledge graphs [8].

The traditional machine learning approach – where the features had to be manually created – has advanced significantly with the introduction of sophisticated auto-learning technologies like deep learning, embracing convolutional neural networks (CNN) [9], recurrent neural networks (RNN) [10], and autoencoders. These technologies demonstrated exceptional performance in high-dimensional data processing tasks such as image analysis, video processing, audio recognition, movement detection, and natural language processing [11]. While deep learning effectively captures latent representations of Euclidean data, there are an increasing number of applications where data are represented in the form of graphs. For example, e-commerce, social networks, citation networks, and communication networks. In our context, graphs represent entities and interactions between these entities and do not possess Euclidean structure to be represented through spatial proximity. The success of the convolutional approach in deep learning strategies continued to captivate researchers, who developed a convolutional technique appropriate for graph structures [12]. Fig. 1, wonderfully depicts how convolution operations of the CNN model be applied to graphs. This technique involved aggregating features from immediate neighbors and using message passing to spread this information, resulting in the renowned Graph Neural Network (GNN) [13] or Graph Convolutional Network (GCN) [14].

Combining cutting-edge, promising technologies from the fields of quantum computing and classical machine learning has been the focus of much recent research. Gradual progress has been made in this area, beginning with Quantum Graph Neural Networks QGNN [15,16], which inherit traits from RecGNN [17] and acquire the target node representation by transmitting neighborhood information iteratively until a stable fixed point is identified. Additionally, convolutional neural networks (CNNs) were developed to apply convolution to non-Euclidean spaces, such as graphs, after CNNs were successful in image-related machine learning tasks [18]. Later, a few works in the field [19,20] suggested combining Graph Neural Networks with Quantum Computing to create the QuGCN model. However, these techniques required a cycle of full quantum projection, demanding a large number of qubits, which might not be possible in the present NISQ era [21,22]. Several hybrid quantum–classical models have been devised to circumvent this [23]. One such study by Liu et al. [24] specifically attempted to create hybrid quantum enhancements using a classical convolution neural network, but they were designed to convolute spectral or spatial information obtained from data like images and videos and are not suited for the non-Euclidean graph data. Therefore, there is still room to investigate hybrid GCN and quantum technology integration for graphs and network science. In order to get the benefit of a promising graph convolutional network and to obtain quantum benefits with the least amount of quantum resources possible, we suggest a novel hybrid quantum–classical Graph Convolutional Network-Based Link Prediction (QGCNLP).

The fundamental tactic of our novel QGCNLP approach is to retrieve the node embeddings in the same manner as conventional GCNs and apply convolution by aggregating and passing messages. And, once we have the convolutional features, apply quantum projection to them and instantly measure them down to classical space to produce the activation. This optimal method enables the utilization of graph convolution and eliminates the limitations of restricted qubit circuits. The primary motivations behind the works are:

- To the best of our knowledge, the hybrid quantum approach for convolutional graph neural networks, i.e., hybrid quantum enhancements to convolution on non-spatial graph data, has not yet been investigated by any of the current techniques.
- In network research, deep learning techniques such as GCN and GNN have demonstrated exceptional performance, and they reduces the overhead of feature generation by just requiring raw graph data.
- The quantum optimization applied while convolving on the aggregated feature provides the ideal conditions, as with few quantum resources, it takes advantage of quantum high-dimensional projection and superposition, which enhances the activation accuracy.
- Networks possess complex correlations; Quantum hybridization, due to its high-dimensional, non-linear feature interactions, captures greater expressibility. Moreover, link prediction faces severe class imbalance and missing-edge bias, quantum circuits naturally regularize the model for improved generalization.
- Quantum-optimized deep learning could accelerate exponentially using quantum parallelization.

With these motivations, our article not only makes use of earlier mentioned cutting-edge technologies for performance improvement but also sets a foundation for the effective use of tiny quantum circuits with exceptional performance. The major contributions of our work include:

- Our proposed *QGCNLP* model presents a novel and efficient hybrid integration of two cutting-edge technologies – graph convolution and quantum computing – unlocking a new frontier in intelligent network analysis.
- We employ two successive quantum-optimized convolution layers where the aggregated weights obtained after convolution are projected to quantum space and then measured back to classical values to generate better activation values for the next layer.
- In contrast to earlier approaches [19,20], which call for the entire quantum machine learning cycle that necessitates input data points mapping to quantum qubits and, as a result, a high qubit count and quantum complexity in the present NISQ era. Our approach uses hybrid quantum optimization, applied only to the convolution layer features, producing remarkable results with a small number of qubits.
- We successfully demonstrate that the performance of *QGCNLP* is higher than that of its classical counterparts, i.e., GCN, thanks to the quantum-enhanced convolution.
- Through rigorous experimentation, we have shown that our proposed model *QGCNLP* performs better than a number of state-of-the-art (SOTA) approaches as well as classical link prediction benchmarks. We also establish the model's real-world scalability on larger and dynamic networks.

We determine the ideal performance environment or hyperparameters for our approach by varying the number of qubits, learning ratios, circuit patterns, and evaluation metrics. We also choose the most relevant and nearest GNN technologies for comparison of our model, including its classical counterpart GCN [26], GraphSage [27], and Graph Attention Networks (GAT) [28] and establish that our model surpasses these technologies in terms of performance over several metrics. We perform scalability analysis to test the model's scalability on real-world large and dynamic scenarios.

Organization. Section 2 outlines the Preliminaries and the underlying technologies. Section 3 details the related work in the field; Section 4 contains the proposed approach; Section 5 contains an extensive list of experimentation performed; and Section 6 performs the scalability analysis to challenge the model on larger and dynamic datasets. Lastly, Section 7 presents the execution efficiency and complexity comparisons, and Section 8 provides conclusions and future work.

2. Preliminaries

Since link prediction is the research's application domain, some information on the methodologies utilized to implement the concept is provided. Given that our suggested method depends on two prime technologies, in order to better grasp the proposed methodology, we first explain two domains: the classical graph neural networks (GNN) and their variations and quantum technologies. Convolutional neural networks' success served as an inspiration for the creation of GNNs, which is why they are also included here.

2.1. Link prediction strategies

A range of link prediction points of view for different kinds of graphs have been included in studies. Similarity-based, probabilistic, domain/semantic similarity-based, stochastic block models, and matrix factorization models comprise the fundamental framework for link prediction techniques [29,30]. The idea of moving a network via a stochastic process is used in link prediction using random walk techniques to calculate the probability of future links between nodes [31]. According to the Deepwalk [32] methodology, this method is predicated on the notion that nodes that are frequently visited together during random walks have a higher probability of establishing a link. Random walks are used by methods such as Personalized PageRank [33] and the Node2Vec algorithm [34] to capture local and global network structures, respectively. Other well-known methods used to improve link prediction performance include integrating auto-learners such as recurring neural networks (RNNs) [35], autoencoders [36], and deep learning [37,38]. Finally, this section will discuss the graph equivalents of these technologies, i.e. GNN and GCN [39], which have been utilized in our study.

2.2. Classical auto-learners

Deep learning and other powerful auto-learning technologies have revolutionized artificial intelligence. By allowing computers to learn from their own generated characteristics instead of depending on manually engineered inputs, these techniques enable machines to analyze and comprehend complex data patterns. Techniques like Convolutional Neural Networks (CNNs) [40] are crucial for image recognition, medical imaging, and video analysis because of their exceptional capacity to process visual data. Because of their ability to process sequential data, recurrent neural networks (RNNs) [41] – in particular, Long Short-Term Memory (LSTM) variants [42] – are utilized in time series analysis, speech recognition, and natural language processing. Auto-encoders, on the other hand, uncover meaningful representations by compressing and reconstructing data, discovering latent features critical for tasks like anomaly detection or noise reduction [43]. These techniques have been extensively used for network analysis tasks like node classification, link prediction, community detection, hub/centrality identification, and influence maximization [44–47].

2.2.1. Graph neural networks (GNN)

The success of CNNs motivated extending convolutional principles to networked data. However, unlike images, graphs lack inherent spatial locality; hence, they rely instead on topological neighborhoods to define meaningful local patterns. Graph autoencoders like GNNs and GCNs solve this problem by combining and converting input from the node's surroundings to perform convolution on graph data.

A GNN is a form of artificial neural network that works with data that may be represented as a graph by exploiting pairwise message passing, which allows graph nodes to iteratively update their representations by exchanging information with their neighbors. The feature or information aggregation that follows message transfer is typically the weighted average of the node's own features and the information from its neighbors. As introduced by Scarselli et al. [48], the message-passing equation in graph neural networks can be expressed as:

$$h_i^{(l+1)} = \text{Update} \left(h_i^{(l)}, \text{Aggregate} \left(\{m_{ij}^{(l)} \mid j \in N(i)\} \right) \right) \quad (1)$$

Here, $h_i^{(l)}$: Feature vector of node i at layer l , $N(i)$: Set of neighbors of node i , $m_{ij}^{(l)}$: Message sent from node j to node i at layer l . The Aggregate($\cdot$) function aggregates messages from neighbors (e.g., sum, mean, or max), and the Update($\cdot$) function updates the node's features based on the aggregated message. Several variants of GNN model have been created, starting with recursive or convolutional constructive approaches to perform various types of message forwarding.

One such methodology is the DGCN model, a time-aware graph convolutional framework [49] that directly models the temporal evolution of nodes and edges, which is proposed for representation learning on dynamic graphs. The method improves performance on downstream tasks like node categorization and link prediction in evolving networks by including time-aware mechanisms into GCNs, which capture both structural interdependence and temporal dynamics. CFLP, a counterfactual learning framework [50] for link prediction, is a model that generates hypothetical (counterfactual) links in order to better comprehend causal interactions in graphs. The model enhances resilience and generalization by comparing observed and counterfactual edges, resulting in more precise link prediction under distribution shifts.

2.2.2. Graph convolutional networks (GCN)

GCNs represent a class of GNNs that involve convolutional operations. It is named so because of the shared filter parameters across all node locations [26].

For graph-level outputs, pooling operations are introduced, as discussed in [51]. The connections between nodes in graph data add another level of intricacy. Since networks are widely believed to display network homophily, these linkages are important. Similar nodes are more likely to be connected to one another than dissimilar ones, according to homophily. We can enhance our representation of a node by merging its features with those of its neighbors. This procedure is referred to as convolution or neighborhood aggregation, and it can be shown as:

$$h_i = \sum_{j \in \tilde{N}_i} W x_j \quad (2)$$

where the neighborhood of node i , including itself, is represented by the symbol $\tilde{N}_i$. Since the degree of the neighbors must be taken into account, normalization is also necessary for graph convolution. Researchers discovered that nodes with many neighbors transmit their traits much more easily than nodes with fewer neighbors. They suggested that in order to prevent this impact and balance the influence across all nodes, traits from nodes with fewer neighbors should be given bigger weights as follows:

$$h_i = \sum_{j \in \tilde{N}_i} \frac{1}{\sqrt{\deg(i)}\sqrt{\deg(j)}} W x_j \quad (3)$$

The objectives stated before and formulated in Eqs. (2) and (3) were first proposed by Kipf et al. [26], and the authors emphasized that each layer of GCN, i.e., Graph Convolution Layer (GCL), can be expressed as a non-linear function:

$$H^{(l+1)} = f(H^{(l)}, A) \quad (4)$$

with $H^{(0)} = X$ and $H^{(L)} = Z$ (or z for graph-level outputs), where Z represents the l th layer features, L is the total number of layers, A is the adjacency matrix. Variations in GCN models arise from the specific design and parametrization of the function $f(\cdot)$. The theoretical formulation for the function $f(\cdot)$ for a specific graph-based neural network model $f(X, A)$ is given below. We will use this formula in the rest of this paper and further quantum enhancements. We consider a multi-layer GCN with the following layer-wise propagation rule:

$$H^{(l+1)} = \sigma \left(D^{-\frac{1}{2}} \hat{A} D^{-\frac{1}{2}} H^{(l)} W^{(l)} \right). \quad (5)$$

Here, $\hat{A} = A + I_N$ is the adjacency matrix of the undirected graph G with added self-connections. I_N is the identity matrix, D is the degree matrix and $\hat{D}_{ii} = \sum_j \hat{A}_{ij}$, and $W^{(l)}$ is a layer-specific trainable weight matrix. $\sigma(\cdot)$ denotes a non-linear activation function, such as the ReLU. $H^{(l)} \in \mathbb{R}^{N \times D}$ is the matrix of activations in the l th layer, $H^{(0)} = X$.

The weights are optimized and adjusted by repeatedly iterating the GCN network layers. Here, we can see that Eq. (5) has also accomplished the convolution operation and the necessary degree normalization for the source and the neighboring nodes.

2.3. Quantum computing basics

Quantum computing is an allied discipline that applies quantum physics principles to the computational domain [52]. The characteristics like superposition, high-dimensional Hilbert spaces, and entanglement – achieved through laser pulses or trapped ions and similar techniques in quantum physics – are utilized in enhancing the data expressiveness in quantum computing. This is achieved through quantum circuits and quantum gates in quantum computing. This article explains the fundamentals of quantum computing.

The basic unit in quantum computing, the quantum bit, is called a qubit. It can be represented as a linear combination of two basic states:

$$|\phi\rangle = a|0\rangle + b|1\rangle \quad (6)$$

where $|0\rangle$ and $|1\rangle$ are the basis states. a and b are known as amplitudes, which are complex numbers that satisfy $a^2 + b^2 = 1$.

For an n -qubit system, a vector, $\mathbf{x}$ with 2^n complex elements can be used to represent the amplitudes of 2^n quantum states. All the elements in $\mathbf{x}$ satisfy:

$$\sum_{i=0}^{2^n} |x_i|^2 = 1 \quad (7)$$

A set of qubits forms a quantum circuit, where computation involves transitioning the qubits from one state to another. The fundamental building block of this process is the quantum gate. Each quantum gate can be expressed as a unitary matrix, $U(\theta)$, with θ being a parameter that can be adjusted during training. Quantum computation, therefore, entails altering the states of the qubits, such as:

$$|\phi\rangle = \dots U(\theta) |\phi_0\rangle \quad (8)$$

where $|\phi_0\rangle$ is the initial state, and what $|\phi\rangle$ is the final state? In a quantum circuit, a set of quantum gates is performed sequentially to represent a function. This circuit's computational values, which quantify the likelihood of a qubit state, such as the state $|0\rangle$ for Z measurement, are extracted at the end using a readout circuit. There are various methods for converting classical data to quantum data, including amplitude encoding and angle encoding. In order to encode the data, $U(x)$. The qubits are entangled in the compute unit W to represent a function that has a collection of trainable parameters. The following sub-sections highlight key concepts in quantum computing that underpin its claimed supremacy.

2.3.1. Quantum entanglement, superposition and Hilbert space

Superposition refers to the principle that any quantum state can be represented as a linear combination of a few basis states. For example, a single qubit can be described as a combination of $|0\rangle$ and $|1\rangle$. Similarly, the state of a system with n qubits can be expressed using 2^n bit-string states, which are created through the tensor products of $|0\rangle$ and $|1\rangle$. These 2^n bit-string states form an orthonormal basis known as the computational basis.

System states that cannot be represented as a tensor product of its individual subsystem's states are known as entangled states. It implies that the individual states of subsystems do not exist independently anymore. Instead, the system must be described by the probabilities of all possible combinations of the qubits. For an n -qubit system, this means accounting for the probabilities of all 2^n states. This property makes quantum circuits challenging to simulate on classical computers. However, it also highlights the extraordinary potential of quantum computing [53].

By passing a single qubit through one logic gate, a quantum circuit with n qubits can perform 2^{n-1} calculations simultaneously. For instance, in a system of 100 qubits, passing a single qubit through a single gate enables 2^{99} calculations. This demonstrates exponential speedup [54]. Thus, entanglement allows computations to leverage a 2^n -dimensional complex vector space using only n physical qubits.

A Hilbert Space (H) is a specific type of linear vector space that includes additional properties beyond those of a standard linear vector space [55]. One key feature is the inner product operation, which measures the separation or relationship between two vectors. This inner product is defined as $\langle v_1, v_2 \rangle \in \mathbb{C}$, the set of complex numbers. Hilbert Space provides the benefit of a high-dimensional complex vector space, which makes it suitable for representing two-state quantum systems.

2.3.2. Quantum data projection

As initially proposed by Havlicek et al. [56], the set of rotations, and entanglements applied to qubits yields a circuit that is difficult to simulate classically. The classical data is projected into a complex quantum vector space using a parameterized feature map circuit, which is mathematically represented as $U_{\phi}(x)$:

$$U_{\phi}(x) = \prod_d \bigcup_{\phi(x)} R_Y^{\otimes n} \quad (9)$$

Here, n represents the number of qubits and d refers to the circuit depth, i.e., the number of repetitions of the following components:

1. The Rotation gate (R_Y), and
2. The entangling block ($U_{\phi(x)}$).

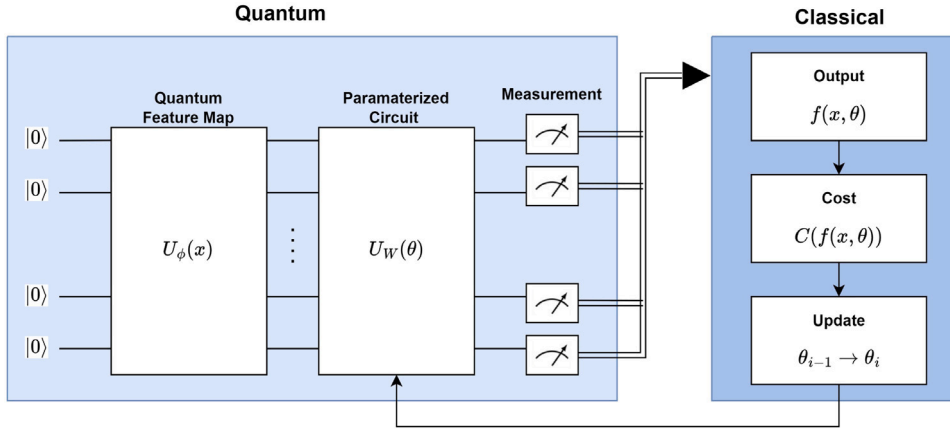


Fig. 2. Scheme of a hybrid quantum–classical VQC presented by De Falco et al. (2024) [57].

2.3.3. Quantum machine learning (QML)

The core of Quantum Machine Learning (QML) is the concept of Variational Quantum Algorithms (VQA) that map the classical features to qubits and perform feature learning in the quantum space, exploiting quantum benefits like superposition, high-dimensional Hilbert space, and entanglement to generate the circuits that are too complex to simulate classically. Quantum circuits essentially include data encoding, which is mapping classical data into quantum space, achieved using superposition through Hadamard gates and entanglement via CNOT gates. The process then includes training the data with quantum optimization methods and parameter shift techniques. Finally, quantum measurements leverage observables to translate quantum probabilities into classical outcomes. This section delves into the technical details of each stage. These phases of VQA are demonstrated in the first block of Fig. 2.

Variational/Parameterized Quantum Circuit (VQC/PQC) [58,59] performs the core operation of QML i.e. quantum feature learning and optimizations. They are commonly known as Ansatz. They exploit a quantum feature map like the data encoding, but the parameters are variational and are updated with quantum parameter shift techniques through the training iterations.

3. Related work

The field of quantum exploration in neural networks and graph neural networks is rather well-established, despite the fact that the quantum hybridization of graph convolutional networks is still unexplored. Despite being a relatively new field of study, a great deal of work has been done in this area. Here, we go over a few of the most important and significant works that form the foundation of our investigation.

3.1. Quantum graph neural networks (QGNN)

In the realm of QML, the definition of a Quantum Graph Neural Network (QGNN) that learns and makes predictions about the underlying graph by using the Ansatz of a quantum circuit is regarded as a groundbreaking technique. The fundamental concept is to use quantum dynamics to embed data and create a richer feature map with properties that are difficult for conventional approaches to access [60]. Complete utilization of Quantum computing capabilities in all the Machine learning phases was exploited by Singh et al. in [61], where they proposed Quantum Neural Network dealing with performing data projection, neural network-based quantum training, and quantum measurements all in quantum space. Another use of quantum data projection capabilities was efficiently explored in the form of Projected Quantum Kernel and applied to random walk embeddings, giving rise to the PQKELP [62] model for link prediction in dynamic networks.

In particular, the Hamiltonian of the system in a QGNN encodes the graph topology in the dynamics, suggesting that quantum circuits with graph-theoretic characteristics might be imagined [63]. Quadratic Hamiltonians may therefore be converted into unitaries (quantum circuits) in a clean manner while maintaining the same connection to a graph. Generally speaking, it is exceedingly challenging to implement this type of unitary on a quantum computer that describes the entire circuit. However, Jones et al. [64] have demonstrated that it may be approximated using the Trotter–Suzuki decomposition. In this way, QGNNs are designed to be VQCs that may use a variational method to learn the system’s underlying dynamics.

In 2019, Verdon et al. introduced QGNNs [65]. The idea of Networked Quantum Systems was presented by the authors. They look at a graph with the formula $G = (V, E)$, where E is the graph’s set of edges and V is its set of nodes. This viewpoint allows one to designate a quantum subsystem with Hilbert space H_v for each node $v \in V$ in the graph, creating a global Hilbert space $H_V \equiv \bigotimes_{v \in V} H_v$, where each node or subgraph represents one or more qubits. A quantum network is a configuration in which the graph’s edges control communication between the sub-spaces of the nodes [66,67]: couplings between degrees of freedom on two distinct nodes are permitted if an edge between them exists.

A novel combination of GNNs with quantum computing, quantum spatial-temporal GNNs are designed especially to manage the complexities of spatial-temporal data. Qu et al. [68] introduced the approach for Quantum Spatial-Temporal GNNs, which starts with the use of quantum circuits to convert classical input into quantum states. A temporal analysis based on the Schrödinger technique follows this transformation, in which the dynamics of the data are represented as quantum walks that depict the change of quantum states over time.

3.2. Quantum graph convolutional networks (QGCN)

A QConvGNN model including state preparation, quantum graph convolution, quantum pooling, and quantum measurements are put out by Zheng et al. [20]. In this case, the connection is represented by $|0\rangle$ or $|1\rangle$ on the node-pair-representing qubits, and node features are recorded into a quantum register per node. Pooling and convolution are implemented as measurement-conditioned unitary operations and controlled two-node unitaries, respectively.

Zhang et al. present a QConvGNN model for non-Euclidean data that is applied to PQC in a different study [69]. This model comprises network optimization, quantum encoding, a quantum graph convolutional layer, and a quantum graph pooling layer. Classical feature vectors are transformed into quantum states via quantum amplitude encoding.

Integrating graph topology information into quantum circuits while preserving the learnable characteristics that define conventional GCNs is the major issue that QuGCN attempts to solve. VQCs manage learnable parameters, while “Givens rotations” [19] encode graph information.

When amplitude encoding is used to encode node attributes, as in Equation, quantum resource needs are greatly reduced in comparison to approaches that require $O(N)$ or more qubits. With node feature propagation comparable to the layer-wise processes of traditional GCNs, the QuGCN framework incorporates VQCs for learnable parameters.

3.3. State-of-the-art models

As was covered in the preliminary section, GNNs are neural networks that have been specially designed to carry out tasks connected to graphs. Certain neural networks, like GCN, make use of convolutional capabilities. In this section, we outline some of the most popular and effective variations of this technology that are used for link prediction and other network activities. Most of the GNNs perform the neighborhood feature aggregation in two main steps: (1) the message passing step and (2) the message aggregation step. This can be mathematically stated as follows:

$$h_v^{(l)} = AGG^{(l)}(\{m_u^{(l)}, u \in N(v)\}) \quad (10)$$

Here, the first part of the equation $AGG^{(l)}$ formulates the message aggregation step, and it can be any of these aggregation function $Sum(\cdot)$, $Mean(\cdot)$, $Max(\cdot)$. $N(v)$ represents the set of neighbors for node v .

The second part $\{m_u^{(l)}, u \in N(v)\}$ is the message passing step, and it can be any activation layer, for example, a linear layer $m_u^{(l)} = W^{(l)}h_u^{(l-1)}$, where $W^{(l)}$ the NN layer weights.

All the other graph methods, including GCN perform these two steps in some form or the other [70,71]. Some of the state-of-the-art GNN-based methods are described here.

- Graph Convolutional Network (GCN).

It is the classical version of our exact approach, acting as the base model for our methodology, on which we apply quantum enhancement. GCNs employ deep learning models to graphs by “extending the concept of convolutional neural networks (CNNs) to graph-structured data” [72,73]. They aggregate and combine information from neighboring nodes iteratively. Basically when looking at the formulation in Eq. (10) and Eq. (13), we may observe that the aggregation function is $\sum(\cdot)$ and message passing is done with a linear layer of activation. They are much simpler than the other Graph ML methods and have the ability to capture embeddings not only at the node level but also at the graph level.

- GraphSAGE.

By sampling a fixed number of neighbors for feature aggregation, GraphSAGE becomes suitable for larger graphs. It is another variant of GNNs where the aggregation function may be mean, max-pooling, or LSTMs. Another advantage of this method is that it ensures that the node’s own features are accounted for in the aggregation; hence, it has two levels of aggregation, making it more inductive. The model is more capable of predicting unseen nodes.

- Graph Attention Networks.

GATs enhance GCNs by assigning different attention weights to neighbors, emphasizing more important connections. The term “attention” [74] gives relative weights between two connected nodes. It measures the connective strength between the node u and its neighbor v as:

$$a_{u,v} = \text{softmax}(\text{LeakyReLU}(a^T[W_u || W_v])) \quad (11)$$

Since it considers the fact that the effect of each node is neither similar to the GraphSage nor predetermined like the GCN, it is more adaptive to realistic networks. It also works well on heterogeneous networks [75].

In order to benchmark our model against its closest counterparts, recent approaches based on Quantum-GCNs are also incorporated into the state-of-the-art comparison, and a brief outline for these models is presented here.

- Quantum Spatial Graph Convolutional Network (QSGCN) [69], a quantum neural network called the QSGCN was created to handle non-Euclidean graph-structured data directly on parameterized quantum circuits (PQCs) that are compatible with NISQ devices. It uses controlled quantum gates to achieve graph convolution and pooling, and it transfers vertex characteristics and adjacency information into quantum states. The parameter-shift rule and quantum stochastic gradient descent are used in a hybrid quantum–classical optimization strategy to train the model.
- Graph Quantum Walk Neural Network (GQWNN) [76], a novel graph learning framework that addresses the absence of graph inductive bias in plain self-attention models by incorporating quantum walk-based structural encoding into graph transformers. In order to produce learnable structural encodings that simultaneously capture graph topology and node characteristics, GQWNN uses attribute-aware discrete-time quantum walks. These encodings are then injected as bias terms into the transformer’s self-attention mechanism for more significant global interactions. The sequential and temporal dependencies of quantum walk trajectories are exploited by a quantum walk recurrent module, which is used to supplement this global modeling and increase the model’s sensitivity to local structural patterns.

4. Proposed work

Given all of the earlier techniques described in related work, such as QGNNs, QrecGNNs, QuConvGNNs, etc., it is easy to see that these techniques have tried to use quantum enhancement in a full-fledged learning process by either monopolizing VQCs for parameter tuning or assigning a quantum Hilbert space at each node or edge level, meaning that each node or edge represents one qubit. As has been already stated, this is computationally intensive and may not be feasible in the current NISQ era. On the other hand, our method aims for a hybrid quantum–classical strategy that combines quantum advantages with the core of classical graph convolution neural network-based machine learning. Fig. 2 depicts the standard stages of a variational quantum circuit, whereas Fig. 3 provides a schematic overview of the architecture and operational phases of the proposed method. In order to apply the quantum benefits, the features obtained after convolution are projected into quantum space, tuned with VQC, and immediately measured back to the classical state in order to identify the layer activations. This presents an effective hybrid quantum–classical graph convolutional network while minimizing quantum computation and yielding optimal quantum advantages at a very low computational cost.

4.1. Theoretical explanation

This section explains the formulation and mathematics that underpin our approach. To better comprehend the algorithmic side of the model, a firm grasp of its theoretical foundations is needed. We have modified the classical convolution formulation with quantum modifications, this is shown here in the form of clear and explicit equations.

4.1.1. Classical GCN based graph initialization

As the method follows a classical convolutional neural network, we start with the adjacency matrix and its normalized forms before the learning process. As GCNs specifically use the normalized adjacency matrix to aggregate features, a symmetric normalization is performed in order to balance the influence of a node apart from its degree:

$$\hat{A} = D^{-1/2}(A + I)D^{-1/2} \quad (12)$$

where D is the diagonal degree matrix and A is the adjacency matrix. Note that by adding I to A we add self-loops i.e. the weighted features of the node are counted while taking into account its neighbors.

4.1.2. Build classical GCN layers

The first step was graph generation and normalization. The next step is to build the GCN layer that will be later converted to QGCN layer. A typical GCN layer is described as

$$H^{(l+1)} = \sigma(\hat{A} H^{(l)} W^{(l)}) \quad (13)$$

We employ Eqs. (12) and (13) to yield Eq. (5) as coined in [26]. So the convolution results in aggregated features of each node neighbor, now is the time to apply quantum enhancements to the aggregated features and optimize the activation process.

4.1.3. Quantum optimization for QGCN

To apply quantum circuits to the aggregated features after convolution, we employ the Variational Quantum Algorithm (VQA), introduced by Havlivcek et al. [56]. A variational quantum circuit (VQA) includes a data encoding stage $U_\phi(x)$, a carefully designed parameterized quantum circuit (ansatz) $U_\theta(x)$ applied to the final quantum state, and a measurement operation $f(x, \theta)$ at the end of the circuit.

The data encoding stage $U_\phi(x)$ is the set of unitary transformations constructed from initial gates, including the R_Y rotations and entanglements. This circuit combination produces the following quantum feature map [77].

$$|\phi(x)\rangle = U_{ent} \cdot \bigotimes_{i=1}^n R_Y(\phi_i(x_i))|0\rangle \quad (14)$$

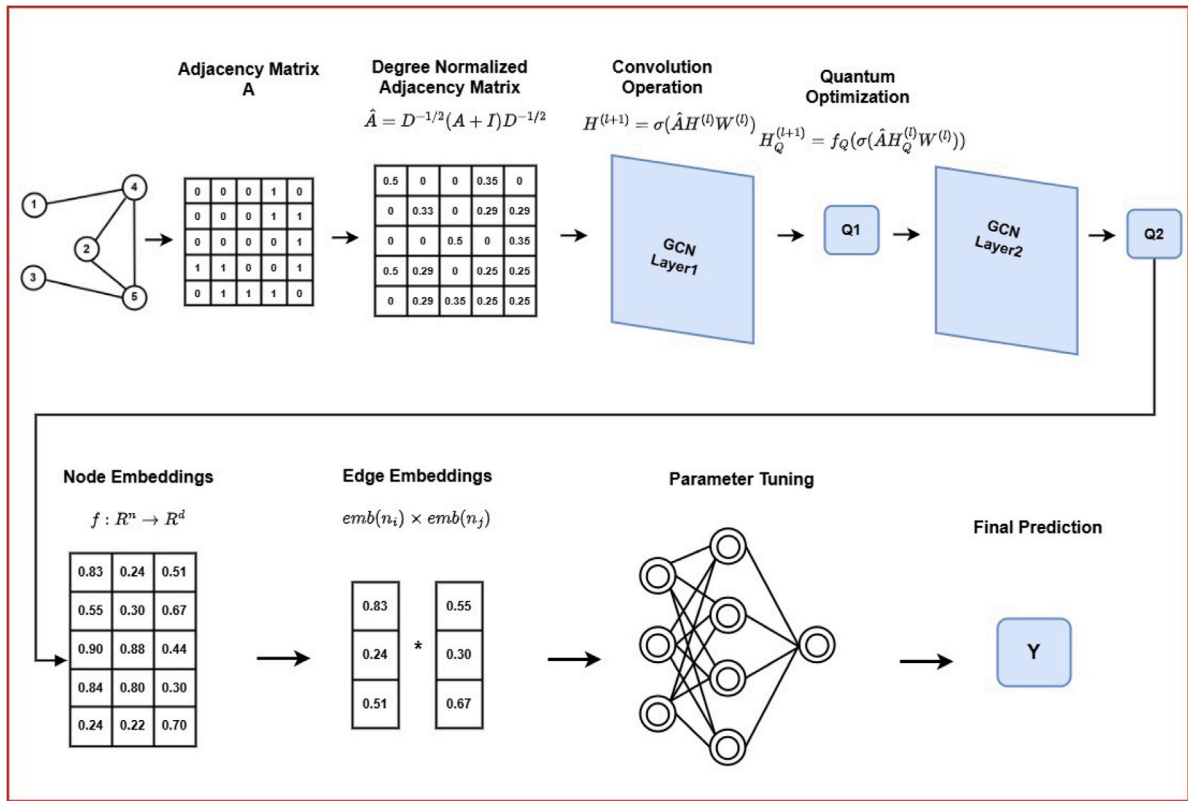


Fig. 3. QGCNLP Flow Diagram. The feature matrix is the degree-normalized adjacency matrix $\hat{A}$ which is then passed through classical GCN layers performing convolutions to achieve aggregated neighborhood features $H^{(l)}$ which is then passed through the quantum circuit to acquire the quantum-optimized features $H_Q^{(l)}$, after passing through the second iteration of these layers, we finally have node embeddings; these are converted to edge embeddings by multiplying the node embeddings. With these edge embeddings, effective loss is computed and propagated backwards for weight optimization, leading to repetitions of GCN and quantum layers until the final readout. Finally, the edge prediction Y is produced.

The parameterized quantum circuit (Ansatz) $U_\theta(x)$ used in our approach is a specific variation of (9), which performs the parameterized learning as follows.

$$U(x, \theta) = U_{ent}(\theta_{ent}) \cdot \bigotimes_{i=1}^n R_Y(\phi_i(x_i) + \theta_i) \quad (15)$$

Here θ is a tunable parameter adjusted with the quantum parameter shift technique. It acts on initial quantum state $\psi(0)$ to produce final state $|\psi(\theta)\rangle = U_\phi(x)|\psi(0)\rangle$ by applying set of parameterized R_Y rotations and entanglement blocks [78]. A measurement operation is carried out at the circuit's end to extract the quantum circuit's result. This forms the objective function of the parameterized circuit:

$$f(\mathbf{x}, \Theta) = \langle \phi(\mathbf{x}) | U_W(\Theta)^\dagger \hat{O} U_W(\Theta) | \phi(\mathbf{x}) \rangle, \quad (16)$$

This is equivalent to estimating the expectation value $\langle O \rangle$ of an observable $\hat{O}$. Several circuit shots are required to obtain the predicted value of the selected observable since quantum physics is probabilistic. After several iterations and Gradient-Based Optimization, the final measurement yields the activations whose accuracy is enhanced with quantum high dimensional Hilbert Space. This eventually changes the classical GCN formulation to QGCN formulation as follows:

$$H_Q^{(l+1)} = f_Q(\sigma(\hat{A} H_Q^{(l)} W^{(l)})) \quad (17)$$

where f_q is the quantum optimization performed with the circuits and techniques formulated in Eqs. (14), (15) and (16). $H_Q^{(l)}$ is the output of previous layers' activation, which is quantum-optimized aggregated (convolutional) node features. Note that $H_Q^{(0)}$ is the input feature map at the beginning, i.e., the input layer, which is basically the adjacency matrix in our case.

4.1.4. Classical training and predictions in QGCNLP

After the quantum enhancement and activation at every layer, the final layer generates the node embeddings. The activation is created by capturing the dot product of these pairwise embeddings and then adding them together. Let $z_i \in \mathbb{R}^d$ and $z_j \in \mathbb{R}^d$

denote the embeddings of nodes i and j , respectively, obtained after the hybrid convolutional layers. The score for the potential edge between them is computed by an inner product decoder:

$$s_{ij} = \langle z_i, z_j \rangle = \sum_{k=1}^d z_{i,k} z_{j,k}. \quad (18)$$

The probability of an edge existing between nodes i and j is given by applying the sigmoid activation.

$$\hat{y}_{ij} = \sigma(s_{ij}) = \frac{1}{1 + e^{-s_{ij}}}, \quad \hat{y}_{ij} \in (0, 1). \quad (19)$$

To train the model, we use both observed (positive) edges $\mathcal{E}^+$ and a set of sampled non-edges (negative edges) $\mathcal{E}^-$. Each pair (i, j) in the training set $\mathcal{T} = \mathcal{E}^+ \cup \mathcal{E}^-$ is associated with a label.

$$y_{ij} = \mathbf{1}\{(i, j) \in \mathcal{E}^+\} \quad (20)$$

With a gradient decent optimizer, this activation value is used to compute loss and propagate backward for cost optimization. The overall training objective is to minimize the binary cross-entropy loss.

$$\mathcal{L}(\Theta) = - \sum_{(i,j) \in \mathcal{T}} \left[y_{ij} \log \hat{y}_{ij} + (1 - y_{ij}) \log(1 - \hat{y}_{ij}) \right], \quad (21)$$

where Θ denotes all learnable parameters of the model.

Lastly, node/edge embeddings are evaluated using the trained GCN model, and the model's accuracy is derived from the predictions.

4.2. Algorithm and flowchart

The theoretical section of the proposed QGCNLP method clearly details the mathematical background of the technique. This section deals the algorithmic and programming methodologies used. Fig. 3 describes the overall structure of the approach and the algorithms 1, 2 describes the vital steps of the program execution. We also mention the tools and libraries along with significant parameters and hyperparameters.

In algorithms 1, line 1–5, initialize the degree normalized adjacency matrix A_{norm} and the adjacency matrix as the feature matrix. In line 6 the training parameters classical as well as quantum are randomly initialized. Now we set up the model. In line 7, it starts with creating a basic neural network “net”. “net” is topped with two convolutional layer $qgcNLayer1$ and $qgcNLayer2$, stacked over each other in line 8–10. The $createQGCNLayer$ performs the actual convolution and moreover, the convolution layers are quantum optimized here in lines 20–24. Line 21 performs the convolution i.e. aggregation of neighborhood features in the case of graphs by multiplying the features with the normalized adjacency matrix and the network weights of the current layer. Line 22 calls the quantum projection circuit to perform quantum enhancement on aggregated neighborhood features which is encapsulated in the algorithm 2.

In the algorithm 2, firstly, the dimension of the intermediate features are reduced to the number of qubits we have in our model. Now for each node's reduced features of size k , we initialize k qubit to state $|0\rangle$ in line 4. In the next two lines, we perform quantum encoding of the data by performing Y rotations. In lines 7–8, we perform qubit entanglement by using $CNOT$ gates. The parameter tuning with a parameterized quantum circuit (PQC) is performed in lines 9–10. Finally, the qubits are measured on a computational basis and converted to classical values in lines 11–12. Finally, the quantum-enhanced values are returned to the algorithm 1 for further processing.

Coming back to the algorithm 1, the final activation with the Relu function is calculated for the quantum to enhance values returned by the *QuantumProjection* function, in line 23. Lastly, in lines 11–16, multiple epochs of training are performed with balanced true and false edge samples. Here the features are passed through repeated layers of convolution and quantum projection, optimizing weights based on loss calculation. The forward pass takes place in line 14. loss calculated in line 15 propagates back to the network in line 16 of the algorithm. The model is trained until the loss is minimized or till it is stable. Finally, in lines 17–19, the prediction values $\hat{Y}_{test}$ are calculated and evaluation metrics for test samples are calculated with true Y_{test} values.

4.3. Libraries and configurations

We have simulated the traditional neural network using the “nn” library module from the well-known *Pytorch* library. We add the convolution operation to it by multiplying the feature by the degree-normalized adjacency matrix and applying weights to this. For classical conventional GCN simulation, we employed the “GCNConv” module from the *torch_geometric* library. Whereas for quantum simulation we used quantum circuits generated by *pennylane*'s “qml” library [79]. For converting classical data to quantum state we use a few Y -rotations. For applying the parameterized circuit we use a few *BasicEntanglerLayers* and a number of R_Y rotations. We utilized the *nn.Linear* library since the aggregated features with a bigger dimension had to be lowered to meet the number of qubits available. Metrics including the AUC, Test Accuracy, Precision, Recall, and F1-score are used in the prediction and evaluation process. These are adequate to illustrate the model's overall performance. Our experimented learning rates are 0.05, 0.01, 0.001, and 0.005. Two models – one with four qubits and the other with eight – were used.

Algorithm 1 Graph Convolution with Quantum Projection**Require:** Graph $G = (V, E)$ with $|V| = n$, feature matrix $X \in \mathbb{R}^{n \times d}$, adjacency matrix $A \in \mathbb{R}^{n \times n}$, number of qubits k **Ensure:** Predictions $\hat{Y}$, $\hat{Y}_{test}$ and evaluation metrics

```

1: Normalize adjacency matrix:
2:  $A_{\text{hat}} \leftarrow A + I$  ▷ Add self-loops to include each node itself
3:  $D \leftarrow$  Diagonal degree matrix of  $A_{\text{hat}}$  ▷ Compute degrees of nodes
4:  $D_{\text{inv-sqrt}} \leftarrow D^{-0.5}$  ▷ Scale by inverse square root of degrees
5:  $A_{\text{norm}} \leftarrow D_{\text{inv-sqrt}} A_{\text{hat}} D_{\text{inv-sqrt}}$  ▷ Normalized adjacency matrix for stable propagation
6: Initialize classical weight matrix  $W \in \mathbb{R}^{d \times d'}$  and quantum parameters  $\Theta \in \mathbb{R}^{d \times k \times 3}$ 
7:  $net \leftarrow$  NeuralNetwork()
8:  $qgcNLayer1 \leftarrow \text{createQGCNLayer}(X, W, A_{\text{norm}})$  ▷ Perform classical quantum graph convolution
9:  $qgcNLayer2 \leftarrow \text{createQGCNLayer}(X, W, A_{\text{norm}})$ 
10:  $model \leftarrow net(qgcNLayer1, qgcNLayer2)$ 
11: for  $i \leftarrow 1$  to epochs do
12:   Sample positive and negative edges
13:   while  $model.is\_trained() = \text{true}$  do ▷ Train model using binary cross-entropy loss
14:      $\hat{Y} \leftarrow model.forwardPass()$ 
15:      $loss \leftarrow \text{calculateLoss}(\hat{Y}, Y)$ 
16:      $model \leftarrow model.backwardPass(loss)$ 
17:  $\hat{Y}_{test} = model.predict(Y_{test})$ 
18: Evaluate metrics: Accuracy, AUC, Precision, Recall, F1, AUPR ▷ Assess model performance
19: return metrics
20: function CREATEQGCNLAYER( $X, W, A_{\text{norm}}$ )
21:    $X \leftarrow A_{\text{norm}} \cdot X \cdot W$ 
22:    $X_Q = \text{QuantumProjection}(X, \Theta)$ 
23:    $Y = \text{activation}(X_Q)$ 
24:   return  $Y$ 

```

Algorithm 2 Quantum Projection Algorithm

```

1: function QUANTUMPROJECTION( $X, \Theta$ )
2:    $X_{\text{reduced}} \leftarrow \text{LinearReduction}(X, k)$  ▷ Reduce features to match qubit count
3:   for each node embedding  $X \in X_{\text{reduced}}$  do
4:     Initialize  $k$  qubits in the  $|0\rangle$  state
5:     for  $i \leftarrow 1$  to  $k$  do
6:       Apply  $R_Y(y[i])$  to qubit  $i$  ▷ Encode classical data into quantum states as  $R_Y(y_i)$ 
7:     for each pair of qubits  $(i, j)$  do
8:       Apply  $CNOT(i, j)$  ▷ Introduce entanglement between qubits
9:     for  $i \leftarrow 1$  to  $k$  do
10:      Apply  $R_Y(\Theta[i, 0]), R_X(\Theta[i, 1]), R_Z(\Theta[i, 2])$  to qubit  $i$  ▷ Parameterized quantum gates  $R_Y, R_X, R_Z$ 
11:      Measure each qubit in the computational basis ▷ Obtain measurement results as classical probabilities
12:      Append measurement results  $q$  to  $Q$ 
13:   return  $Q$ 

```

5. Performance analysis

This section describes the experimental setup and the execution plan for evaluating the performance of the proposed model on different runtime environments. Our objective is to figure out the best runtime configuration for the proposed QGCNLP model that can be used to compare with classical counterparts and other State-of-the-art (SOTA) methods. Here, we outline the experimental configuration and the implementation strategy for assessing proposed model's performance in various runtime environments. In order to do this, we experiment with different runtime systems, learning rates, and the number of qubits. We explore a variety of static and dynamic datasets and circuit patterns in order to achieve this. The following subsections describe each of these experimental settings in more detail.

- **Evaluation Metrics** The performance of the proposed model is evaluated using multiple widely adopted metrics, namely accuracy, precision, recall, F1-score, Area Under the ROC Curve (AUC), and Area Under the precision-recall Curve (AUPR). These complementary measures capture different aspects of link prediction performance, ensuring a comprehensive assessment of both ranking quality and classification reliability. The Hits@50 metric that we used in Table 4, is an Open Graph Benchmark

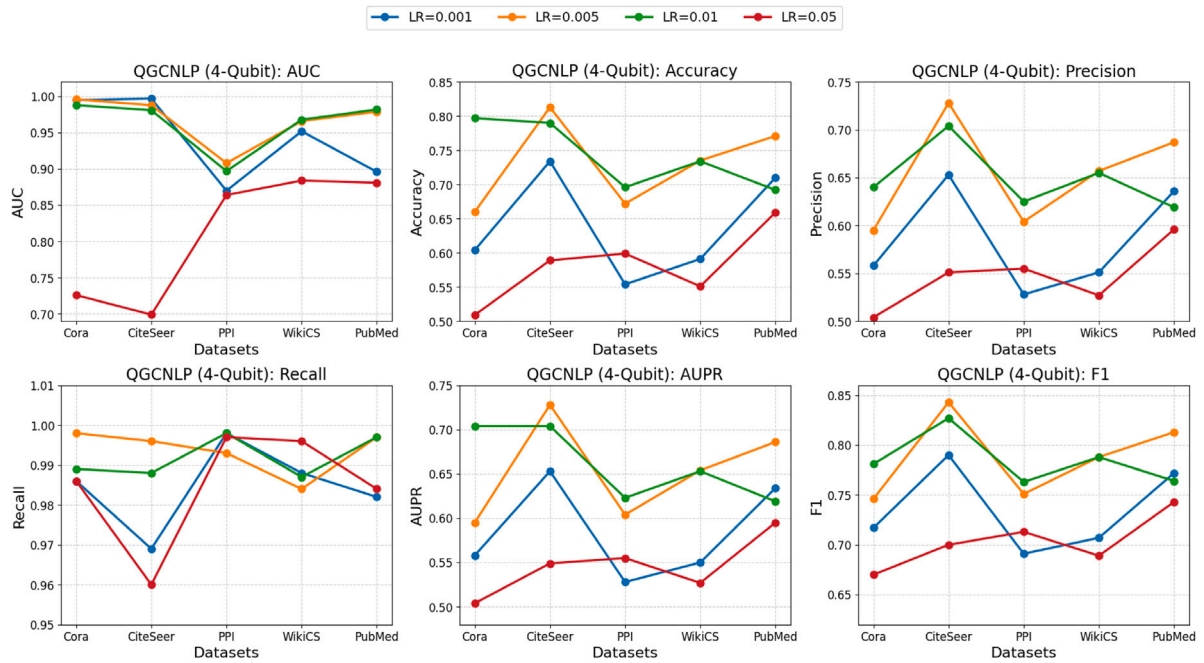


Fig. 4. Performance evaluation of QGCNLP_4qubit model over six performance evaluation metrics, with different learning rates on five datasets.

(OGB) benchmark [80] metric, which is a popular ranking-based evaluation metric commonly used in GCN-based prediction tasks. It calculates the proportion of true positive links that appear within the top 50 predicted candidate edges.

- **Experimental Setup** The first step involves determining the optimal learning rate. We then examine whether the quantum-enhanced QGCN model exhibits training and execution behavior consistent with the classical GCN framework. For this, we have tried four learning rates in the range [0.001, 0.005, 0.01, 0.05]. The availability of a larger number of qubits in current real quantum machines or simulators is the biggest limitation of quantum computational research. T. Häner clearly indicates that “with a 45-qubit entangled circuit simulation, it takes the Cori II supercomputer using 8192 nodes and 0.5 petabytes of memory” [81]. With the server configurations that we had, we could explore till 8 qubits.
- **Datasets** The five standard datasets on which our model was run are extracted from *torch_geometric.datasets* the library. This library hosts datasets representing different network characteristics and belonging to different user domains. Three of them are citation network datasets namely *Cora*, *CiteSeer*, and *PubMed* from the paper [82]. Here in this dataset, nodes represent documents and edges represent citation links. *Cora* has 2708 nodes and 10,556 edges, *CiteSeer* has 3327 nodes and 9104 edges, and *PubMed* is a comparatively larger dataset with 19,717 nodes and 88,648 edges. The fourth one is the protein-protein interaction network *PPI* from the paper [83] where the links represent the connection between protein bonds. It has 2245 nodes and 61,318 edges. The last one is the semi-supervised Wikipedia-based dataset from the paper [84], containing 11,701 nodes and 216,123 edges. Here the links represent the existence of hyperlinks between the Wikipedia pages. The large datasets that we use for comparison in Table 4, are part of the Open Graph Benchmark (OGB), a broad collection of realistic and difficult benchmark datasets designed to support scalable, reliable, and repeatable graph machine learning (ML) research issues. With [169,343 nodes with 2,232,486 edges] and [235,868 nodes with 1,285,465 edges], respectively, they are significantly larger.
- **Dynamic datasets** The dynamic datasets that we use in Table 5, are MIT¹ (human interaction data of MIT students), EU-Core² (email transmission with a European institution), FB-Forum³ (open interactions in a social network), and CollegeMsg⁴ (student engagement on a social media platform). These temporal networks are available on [85], with varying graph sizes, densities, and time periods.
- **Server** The experimentation was carried out efficiently, utilizing high-speed computation capability delivered by the Param Shivay Supercomputer, IIT (BHU), Varanasi. This capability effectively accelerated the quantum circuit simulation of the *PennyLane* library. The convolution operation involving huge matrix multiplication was simplified with the *tensorflow* library and executed on multiple cores of the server node. Jobs were executed on 10 cores of a CPU node (Intel Xeon SKL G-6148 processor (40 cores)).

¹ <http://konect.cc/networks/mit/>

² <https://snap.stanford.edu/data/email-Eu-core.html>

³ <http://networkrepository.com/fb-forum.php>

⁴ <https://snap.stanford.edu/data/CollegeMsg.html>

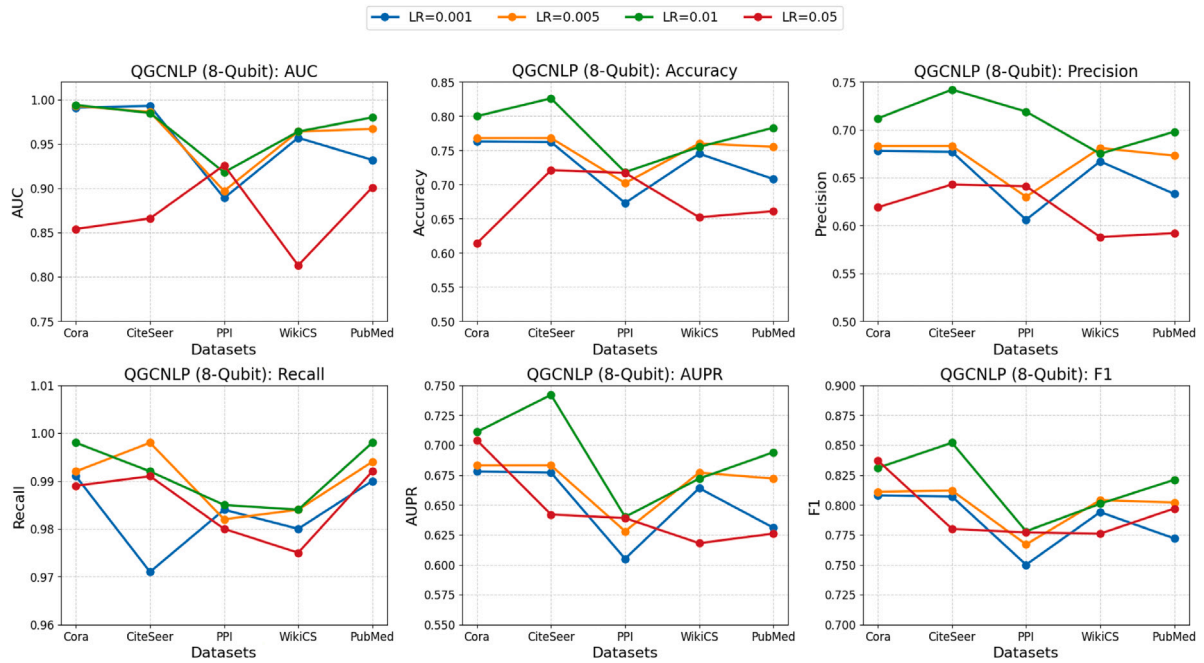


Fig. 5. Performance evaluation of QGCNLP_8qubit model over six performance evaluation metrics, with different learning rates on five datasets.

5.1. Performance evaluation of proposed QGCNLP_4qubit model on different learning rates

This section analyzes the appropriate learning rate for four-qubit quantum circuit simulations, while keeping the test ratio fixed at 0.1 (10%) throughout. Observing Fig. 4, we may find that the AUC scores are highest at a learning rate (LR) of 0.01 for three datasets, namely Cora, WikiCS, and PubMed, and highest at 0.005 on CiteSeer and PPI. For Accuracy, an LR of 0.005 is the highest on CiteSeer, WikiCS, and PubMed, and 0.01 performs well on Cora, PPI, and WikiCS. A similar pattern is observed for the Precision score. For recall, an LR of 0.005 yields the best results on Cora, CiteSeer, and PubMed, whereas 0.01 yields the best results on PPI, WikiCS, and PubMed. Note that an LR of 0.001 also results in a good recall score for the PPI dataset. AUPR and F1 scores also look similar to Accuracy and Precision scores with results distributed among LR of 0.005 on CiteSeer, WikiCS, and PubMed and 0.01 performing well on Cora, PPI, and WikiCS. It can be easily observed that medieval learning rates of 0.005 and 0.01 turn out to be good for our QGCNLP model with 4 qubits.

5.2. Performance evaluation of proposed QGCNLP_8qubit model on different learning rates

This section deals with the appropriate learning rate for eight-qubit quantum circuit simulations, while keeping the test ratio fixed at 0.1 (10%) throughout. On careful observation of Fig. 5, we may find that for AUC, a learning rate (LR) of 0.01 has better performance on four datasets, Cora, PPI, WikiCS, and PubMed, and 0.005 has a competitive score for PPI. LR of 0.001 has performed well on CiteSeer datasets. For Accuracy, an LR of 0.01 has been outstanding for all the datasets. A similar performance is depicted in the Precision score. For recall score, LR of 0.01 performs well on Cora, PPI, WikiCS, and PubMed. Though 0.005 has a competitive score for WikiCS and CiteSeer. In the case of AUPR, similar to accuracy and precision, the LR of 0.01 outperforms all the datasets. Lastly, in the case of the F1 score, LR of 0.01 set the benchmark for Cora, CiteSeer, PPI, and PubMed. Only for WikiCS, 0.005 yields better results. Hence, it may be clearly observed that an LR of 0.01 is the most suitable learning rate for the proposed QGCNLP model. The next section compares the performance of both 4-qubit and 8-qubit models on their individual best LR values; the configuration will be useful for further comparison of the model.

5.3. Performance evaluation of proposed QGCNLP_4qubit model with QGCNLP_8qubit model relative to classical GCN model

This section compares proposed QGCNLP models with various qubit counts to see if there is a positive correlation between qubit counts and performance. Additionally, we would like to find out if adding quantum improvement adds to the performance. Here, we have kept the test ratio fixed to 0.1 i.e. 10%, and the learning rate at 0.01 throughout the evaluation. From Table 1, it can be observed that for the AUC score, the 8-qubit model turns good for three datasets, i.e., CiteSeer, PPI, and PubMed, whereas the 4-qubit model works well for Cora and WikiCS. Only for PubMed, the classical equivalent has a competitive score with an 8-qubit model. Next, the accuracy score QGCNLP_8qubit does well on all datasets except for PPI where the classical GCN model has a

Table 1

Performance comparison between 4-Qubit and 8-Qubit Models, with respect to classical GCN model on five datasets and over six performance evaluation metrics, test-ratio=0.1 & learning rate=0.01.

Metric	Dataset	GCN	QGCNLP_4Qubits	QGCNLP_8Qubits
AUC	Cora	0.967	0.988	0.979
	CiteSeer	0.975	0.981	0.985
	PPI	0.914	0.897	0.918
	WikiCS	0.958	0.968	0.964
	PubMed	0.986	0.982	0.985
Accuracy	Cora	0.719	0.797	0.800
	CiteSeer	0.811	0.791	0.826
	PPI	0.735	0.696	0.818
	WikiCS	0.745	0.734	0.755
	PubMed	0.780	0.692	0.783
Precision	Cora	0.716	0.641	0.712
	CiteSeer	0.726	0.704	0.742
	PPI	0.659	0.625	0.719
	WikiCS	0.676	0.655	0.675
	PubMed	0.695	0.619	0.698
Recall	Cora	0.994	0.989	0.998
	CiteSeer	0.984	0.988	0.992
	PPI	0.976	0.998	0.985
	WikiCS	0.979	0.987	0.984
	PubMed	0.998	0.997	0.994
AUPR	Cora	0.715	0.704	0.711
	CiteSeer	0.726	0.704	0.742
	PPI	0.655	0.623	0.641
	WikiCS	0.662	0.653	0.672
	PubMed	0.697	0.619	0.694
F1	Cora	0.832	0.781	0.831
	CiteSeer	0.841	0.827	0.852
	PPI	0.786	0.763	0.778
	WikiCS	0.800	0.788	0.801
	PubMed	0.819	0.764	0.821

better score. *QGCNLP_8qubit* works well for Precision on all datasets except for Cora, where the classical GCN model has the best result. In case of recall again *QGCNLP_8qubit* works well for three datasets, namely Cora, CiteSeer, and PPI whereas GCN is highest for PubMed and QGCNLP_4qubit highest for WikiCS. AUPR *QGCNLP_8qubit* yields the best results on Cora, CiteSeer, and WikiCS, and the classical model GCN stands well for two datasets, PPI and PubMed. Finally, the F1 score *QGCNLP_8qubit* is best on all datasets except Cora, where the classical GCN model has the best score. With these comparisons, it can be easily deduced that increasing the number of qubits improves the model performance. The 8-qubit model *QGCNLP_8qubit* has better performance over the 4-qubit model *QGCNLP_4qubit*. Also, since the *QGCNLP_8qubit* model has better performance in most of the cases, we may say that the quantum enhancement surpasses the classical GCN based link prediction method.

5.4. Performance analysis for different circuit patterns

In this experiment, we investigate the effect of different quantum circuit configurations on the performance of the link prediction task. This is achieved by applying various rotational quantum gates to introduce amplitude mixing and phase shifts, thereby analyzing their impact on model performance.

In variational quantum circuits, diverse rotation gates play diverse roles in feature transformation. The X, Y, and Z-axis rotations enable nonlinear interactions [86] between encoded node features. Eventually such nonlinearity is essential for link prediction, which necessitates modeling intricate pairwise and higher-order interactions between nodes. The observed improvements in Hits@50, AUC and Accuracy in Table 3 indicate that combining phase encoding with amplitude-mixing rotations enhances relational expressibility, leading to more effective link prediction. Entanglement plays a central role in enhancing the expressive capacity of quantum models [87] for link prediction by enabling non-separable representations of node pairs and their neighborhoods, while increased depth through data re-uploading allows classical features to be injected and nonlinearly transformed across multiple layers [88]. In the context of link prediction, where the task depends on capturing complex pairwise and higher-order node interactions, this combination is particularly effective. The table shows direct proportionality between the circuit complexity and model performance in terms of Hits@50, AUC and Accuracy metrics.

The observed metrics are consistent with the theoretical expectation that increased circuit complexity improves expressivity at the cost of higher computational time. While this limitation is inherent to simulation-based evaluation, it is anticipated to be mitigated on true quantum hardware. For simplicity, we use a basic circuit with Y-axis rotations and linear entanglement, without repetition, to keep computation time minimal for large-scale experiments.

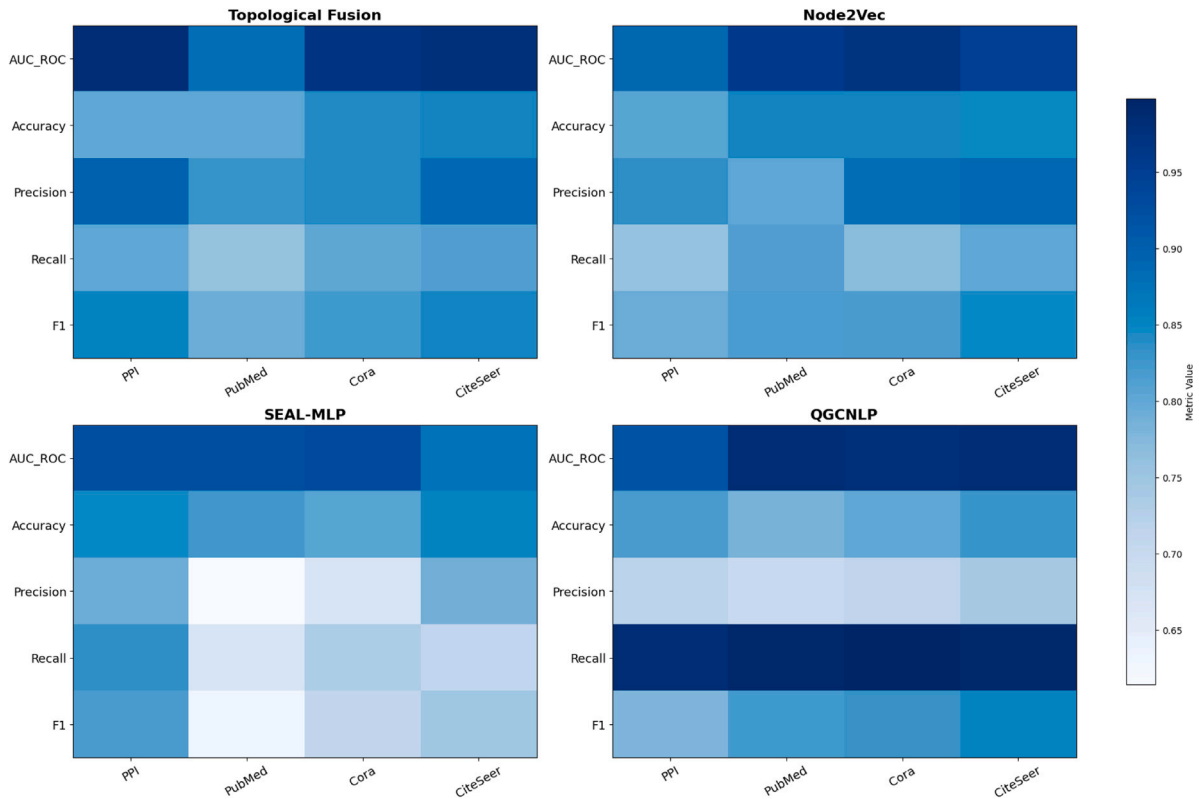


Fig. 6. Performance Comparison of classical benchmark methods with proposed QGCNLP model over different datasets and evaluation metrics.

5.5. Performance evaluation of proposed QGCNLP model with benchmark methods

Since our objective is to assess the improvements introduced by quantum integration in the link prediction task, it is imperative to compare our approach against well-established link prediction methods. Accordingly, an experiment was conducted using classical baselines with the proposed QGCNLP model. Fig. 6 presents the heatmap comparison of three established classical link prediction methods – topological fusion with ML [89], Node2Vec [34], and SEAL-MLP [90] – alongside QGCNLP the model. The results indicate that the proposed model outperforms the baselines in terms of AUC-ROC and recall. While accuracy remains competitive across datasets, highlighting the model's advantages in expressibility, generalization, and robustness. Precision is comparatively lower for certain cases; however, the resulting F1 scores remain strong due to the consistently higher recall.

Overall, this demonstrates that quantum integration with convolution meaningfully contributes to improved representation learning for link prediction, even under practical NISQ-era constraints.

5.6. Performance evaluation of proposed QGCNLP model with state-of-the-art methods

This paragraph evaluates the proposed QGCNLP model using a number of state-of-the-art (SOTA) techniques. Ultimately, this experiment demonstrates if quantum projection offers any advantages over several graph machine learning (GNN) variations that were previously employed for link prediction. Some of these variants are GCN, GraphSAGE, and Graph Attention Network (GAT), explained in Section 3.3. Our strategy and the ground methodologies used to evaluate it must be closely related; hence, we have chosen some of the recent models from the domain of quantum-enhanced link prediction. The models have been carefully picked up to represent the performance of the proposed QGCNLP model with its nearest competitors, which include (1) Quantum Spatial Graph Convolutional Network (QSGCN) [69], a quantum neural network model designed for processing graph-structured (non-Euclidean) data on quantum circuits; and (2) Graph Quantum Walk Neural Network (GQWNN), which leverages attribute-aware quantum walks to generate expressive structural encodings that guide transformer self-attention. These models have been introduced in the related work Section 3.

As the proposed method QGCNLP_{8qubit} performs best with a learning rate of 0.01 and using 8 qubits, we use these configurations for evaluation with state-of-the-art methods. From observing Table 2, it can be derived that the proposed QGCNLP_{8qubit} model gives the best AUC and recall score for all the datasets. For the rest of the metrics, it yields better results for Cora, CiteSeer, and PPI, whereas for WikiCS and PubMed, it GQWNN gives better scores. The nearest competitor is the GQWNN, one shown in Fig. 7, it has considerable computation overhead, especially for these datasets.

Table 2

Performance Comparison of proposed QGCNLP model with state-of-the-art methods, including three classical GCN-based methods and two QGCN-based methods on five datasets and over six performance evaluation metrics, learning rate=0.01, test-ratio=0.1.

Metric	Dataset	Classical GCN methods			Quantum GCN methods		
		GCN	GraphSAGE	GAT	QSGCN	GQWNN	QGCNLP
AUC_ROC	Cora	0.967	0.899	0.897	0.804	0.681	0.979
	CiteSeer	0.975	0.877	0.862	0.824	0.605	0.985
	PPI	0.914	0.825	0.825	0.850	0.908	0.918
	WikiCS	0.958	0.950	0.935	0.927	0.959	0.964
	PubMed	0.986	0.904	0.941	0.924	0.975	0.980
Accuracy	Cora	0.719	0.767	0.755	0.687	0.633	0.800
	CiteSeer	0.811	0.771	0.711	0.707	0.574	0.826
	PPI	0.735	0.698	0.679	0.650	0.809	0.818
	WikiCS	0.745	0.785	0.786	0.708	0.899	0.755
	PubMed	0.781	0.760	0.791	0.743	0.826	0.783
Precision	Cora	0.716	0.706	0.696	0.633	0.627	0.712
	CiteSeer	0.726	0.722	0.658	0.650	0.571	0.742
	PPI	0.659	0.645	0.619	0.593	0.719	0.833
	WikiCS	0.676	0.707	0.712	0.636	0.896	0.675
	PubMed	0.695	0.691	0.712	0.668	0.818	0.698
Recall	Cora	0.994	0.915	0.907	0.888	0.656	0.998
	CiteSeer	0.984	0.884	0.879	0.892	0.589	0.992
	PPI	0.976	0.885	0.929	0.960	0.825	0.985
	WikiCS	0.979	0.974	0.961	0.972	0.903	0.984
	PubMed	0.998	0.946	0.976	0.966	0.936	0.994
AUPR	Cora	0.715	0.688	0.678	0.811	0.653	0.711
	CiteSeer	0.726	0.696	0.639	0.833	0.588	0.742
	PPI	0.655	0.628	0.611	0.858	0.902	0.640
	WikiCS	0.662	0.701	0.704	0.929	0.956	0.672
	PubMed	0.697	0.679	0.707	0.925	0.969	0.694
F1	Cora	0.832	0.797	0.787	0.739	0.641	0.831
	CiteSeer	0.841	0.794	0.753	0.752	0.580	0.852
	PPI	0.786	0.746	0.743	0.733	0.765	0.905
	WikiCS	0.800	0.819	0.818	0.769	0.899	0.801
	PubMed	0.819	0.798	0.823	0.790	0.926	0.821

Table 3

Comparison of prediction performance and execution efficiency of *QGCNLP* model on varying quantum circuit patterns.

Circuit design	Gates	Entanglement	Params	Hits@50	AUC	Accuracy	Exec. time (s)
RY + Basic	RY	Linear	Medium	0.1596	0.9701	0.9241	12.32
RXRYRZ	RXRYRZ	Linear	Medium	0.2239	0.9831	0.9295	13.73
Re-upload $L=2$	RY	Ring	Medium	0.2025	0.9771	0.9221	12.83
Re-upload $L=2$ + RXRY	RX+RY	Ring	Medium	0.2257	0.9789	0.9298	11.44
All-to-all	RXRYRZ	Full	Medium	0.2464	0.9808	0.9269	14.27
RXRYRZ + All-to-all	RXRYRZ	Full	Medium	0.2517	0.9810	0.9316	14.44

Effectively, parameterized quantum circuits represent high-dimensional, non-linear feature interactions and Hybrid GCN + quantum layer captures higher-order structural dependencies. This improves expressibility, which is evident from the higher AUC scores. Moreover Link prediction faces severe class imbalance and missing-edge bias. Quantum-enhanced representations act as implicit regularizers, preventing oversmoothing and overfitting to local neighborhoods. This enhances generalization evident from Improved AUC and Accuracy across multiple datasets. Hence, it can be observed from performance comparison and analysis that our proposed *QGCNLP* model turns out to be more efficient than the quantum as well as the classical counterparts.

6. Scalability analysis

In order to truly evaluate a model's performance on a real network, we must consider real scenarios that are much larger in size and have dynamic behavior, complex relationships, and multiple correlations. These complexities cannot be avoided if the model has to be utilized in practical scenarios. We target these scenarios and evaluate our model on larger datasets to show its performance on growing networks. Then we switch to a few dynamic datasets that show rapidly changing entities and their connections in order to assess its versatility.

Traditional message passing using adjacency matrices is inefficient and hard to scale. The proposed model replaces this with sparse, edge-based message passing that works locally and supports mini-batch training, making it scalable to large and dynamic graphs. This can be realized through the performance evaluations on real-world datasets as shown in [Tables 4 and 5](#).

Table 4

Performance comparison and analysis of proposed *QGCNLP* enhancements to classical GCN models for larger datasets of OGB benchmarks.

Dataset	Model	Hits@50	AUC	AUPR	Accuracy	Precision	Recall	F1 Score
ogbl_collab	QGCNLP	0.3951	0.9731	0.9651	0.9246	0.8959	0.9041	0.9000
	GCNLP	0.1212	0.9640	0.9476	0.8916	0.8096	0.9300	0.8656
ogbn_arxiv	QGCNLP	0.2271	0.9781	0.9726	0.9217	0.8736	0.9860	0.9264
	GCNLP	0.1076	0.9600	0.9486	0.8947	0.8406	0.9742	0.9025

6.1. Scalability analysis on large OGB benchmark datasets

The model has only been tested on small datasets and static networks thus far. Since real networks are huge, an empirical examination of the model's performance on larger networks is required to assess its effectiveness in real networks. Two sizable datasets, ogbl_collab and ogbn_arxiv [80], were used for the experiment. The datasets chosen for large network simulation are mentioned in Section 5.

From Table 4, it is evident that the model performs well on both datasets in terms of accuracy, AUC, and the majority of other metrics. In order to demonstrate the actual performance boost, we have contrasted it with its traditional counterpart, the GCN model. GCN is among the most powerful classical models for link prediction and serves as the classical counterpart to *QGCNLP*. Surpassing GCN, therefore, provides strong evidence that quantum interference remains effective even on large-scale datasets.

6.2. Scalability analysis on dynamic datasets

Dynamic networks, also known as temporal networks, are those that are constantly evolving throughout time. Here, nodes can be added or removed at any moment, and links can be created and deleted. As a result, the edge set and nodes are not fixed. Finding the network's future linkages based on these dynamic characteristics becomes quite difficult in such a situation. We use a reliable timestamp-based graph partitioning method for link prediction. The full graph $G = (V, E)$ is divided into five temporal snapshots. The first four are used for training, and the fifth is predicted using information from the previous ones, then compared with the actual edges. Let $G_1, G_2, \dots, G_n$ denote graphs observed at times $t_1, t_2, \dots, t_n$, the goal is to predict the future graph G_{t+1} .

The model's performance on dynamic networks is shown in Table 5. Two popular models from the classical link prediction domain, namely Node2Vec [34] and LGQ [89], and one quantum-enhanced model, PQCLP [61], that are known to perform better in terms of dynamic link prediction, have been employed for benchmarking. The datasets chosen for dynamic network simulation are mentioned in Section 5.

With Table 5, the model shows strong performance, *QGCNLP* outperforming other methods across most metrics and datasets, indicating good scalability for real-world dynamic networks.

7. Complexity analysis

In the proposed framework, a quantum enhancement is introduced over the aggregated feature representations obtained from the convolutional neural network. While this integration increases the computational complexity relative to the classical baseline, the additional overhead introduced by the quantum component remains marginal. For the quantum enhancement, a small number of qubits are processed using a shallow quantum circuit: first, we perform 'Y' rotations on the qubits; then we entangle the qubits; and lastly, we perform quantum measurements. The complexity of encoding classical data (RY Rotations) [91]:

- The complexity of dimensionality reduction from aggregated features to few qubits is constant $O(1)$
- In general, the complexity of quantum simulator can be assumed to be 2^q where q is the number of qubits
- Overall complexity can be measured as $q * 2^q$, which is $O(2^q)$

Now we can formulate the computational complexity of proposed competitive models. For the general models that apply quantum encoding for graph nodes or the edges, it requires $N(N+1)/2$ qubits. So the total qubits required is $O(N^2)$, where n is the number of nodes. For another category of quantum models for network-related tasks that utilize quantum walks or quantum-inspired walks, the complexity of these walks would be $O(T.N)$ where T denotes the quantum walk steps, generally taken as $T=|\log N|$. Finally, with the simulator, the complexity for both these methods become $O(N^2.2^q)$ and $O(T.N.2^q)$. The comparison of observed execution time per epoch for other models with respect to the proposed *QGCNLP* model is presented in Fig. 7 for better clarity.

Note that the proposed model shows a pattern of least execution time for each epoch among these models, whereas the QSGCN model touches the nearby values. GQWNN has considerable execution time, as for each epoch it performs quantum simulated walks over a set of nodes, which takes considerable time and increases considerably with the size of the network; hence, the spike in the case of the WikiCS dataset.

Table 5

Performance comparison of proposed QGCNLP model with benchmark dynamic link prediction methods, over different dynamic datasets and performance evaluation metrics.

Metric	Dataset	Node2Vec	LGQ	PQCLP	QGCNLP
Test Accuracy	CollegeMsg	0.6357	0.6093	0.6609	0.8095
	Eu-core	0.8752	0.8092	0.8250	0.9092
	fb-forum	0.8645	0.7733	0.7724	0.8893
	mit	0.8814	0.7625	0.7798	0.9391
AUC	CollegeMsg	0.6457	0.4975	0.6627	0.8115
	Eu-core	0.9320	0.7041	0.8226	0.9721
	fb-forum	0.9220	0.6168	0.7660	0.9531
	mit	0.9302	0.7167	0.7802	0.9118
Precision	CollegeMsg	0.6616	0.6295	0.9509	0.9244
	Eu-core	0.8839	0.6281	0.9749	0.9003
	fb-forum	0.8523	0.6164	0.9744	0.8835
	mit	0.9033	0.5233	0.8573	0.9391
Recall	CollegeMsg	0.8136	0.4988	0.3439	0.6741
	Eu-core	0.8700	0.4975	0.6621	0.9003
	fb-forum	0.8847	0.4329	0.5453	0.8970
	mit	0.7624	0.4963	0.6726	0.9794
AUPR	CollegeMsg	0.6651	0.4609	0.7690	0.8730
	Eu-core	0.9318	0.7677	0.8142	0.9661
	fb-forum	0.9263	0.6918	0.7206	0.9472
	mit	0.9105	0.6921	0.7611	0.9405
F1 Score	CollegeMsg	0.5007	0.5170	0.6228	0.7796
	Eu-core	0.8769	0.4502	0.8191	0.9102
	fb-forum	0.8682	0.5164	0.7578	0.8902
	mit	0.8214	0.4829	0.7761	0.9686

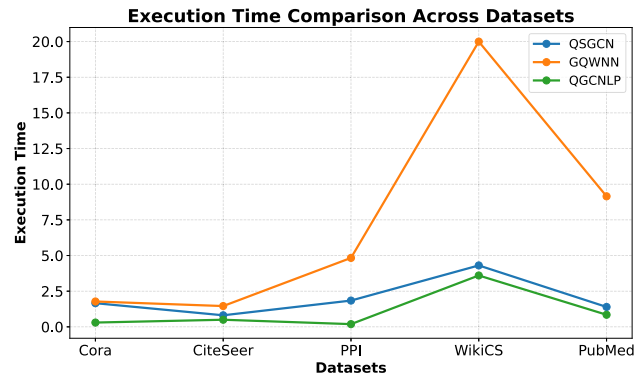


Fig. 7. Comparison of execution time per epoch of quantum graph convolution-based methods with proposed *QGCNLP* model across datasets.

8. Conclusion

The aim of this work was to develop a hybrid classical-quantum method with GCNs for the network-oriented task of link prediction. To address this, we developed the *QGCNLP* model that applies hybrid quantum enhancements to GCN. Unlike the earlier methods that quantize all input features, requiring huge quantum resources in the current NISQ state, the proposed method applies quantum enhancements only to the aggregated features obtained after convolution and message passing. This novel technique yields optimal quantum enhancements in fewer qubits. To prove that the enhancements have resulted in a positive impact on the performance, we compared our model with the most relevant and most recent methods. We started discovering the best configurations and parameters for our model by experimenting with learning rates, circuit patterns, and a varying number of qubits. We found that increasing the number of qubits resulted in improved performance. Through experimentation, we found that the proposed model runs best on a learning rate of 0.01. We also tried different circuit patterns with varying rotation, entanglements, and depths and found that the performance and execution time are directly proportional to circuit complexity. This configuration was used to train and evaluate the proposed model against several state-of-the-art approaches, as well as strong classical link prediction benchmarks, demonstrating its superior performance. From the complexity analysis, we deduce that the quantum enhancements cause minimal quantum overhead. We further conducted a scalability analysis to verify the model's effectiveness on larger datasets and its suitability for real-world dynamic networks, and the results validated its strong scalability.

With all this experimentation and analysis, it is concluded that our proposed *QGCNLP* model improves link prediction performance with minimal quantum resources. These benefits may further be exploited by employing the similar technology to other network learning tasks such as node classification, edge classification, community discovery, and anomaly detection. Additionally, the model can be modified for more intricate, heterogeneous networks with a variety of node and edge types. To further improve scalability, robustness, and predictive performance, deeper quantum circuits, adaptive quantum feature encoding, and hybrid attention methods may be included.

CRediT authorship contribution statement

Nisha Singh: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Conceptualization. **Mukesh Kumar:** Writing – review & editing, Validation, Supervision, Project administration, Data curation. **Jaideep Sharma:** Writing – review & editing, Writing – original draft, Software, Methodology. **Bhaskar Biswas:** Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

“We acknowledge the National Supercomputing Mission (NSM) for providing computing resources of ‘PARAM Shivay’ at the Indian Institute of Technology (BHU), Varanasi, which is implemented by C-DAC and supported by the Ministry of Electronics and Information Technology (MeitY) and the Department of Science and Technology (DST), Government of India”.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at <https://doi.org/10.1016/j.physa.2026.131449>.

Data availability

[Pytorch Geometric Dataset \(Reference data\)](#) (Pytorch Geometric)

References

- [1] F. Xia, K. Sun, S. Yu, A. Aziz, L. Wan, S. Pan, H. Liu, Graph learning: A survey, *IEEE Trans. Artif. Intell.* 2 (2) (2021) 109–127.
- [2] S. Nilizadeh, F. Labrèche, A. Sedighian, A. Zand, J. Fernandez, C. Kruegel, G. Stringhini, G. Vigna, Poised: Spotting twitter spam off the beaten paths, in: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1159–1174.
- [3] M. Grbovic, V. Radosavljevic, N. Djuric, N. Bhamidipati, J. Savla, V. Bhagwan, D. Sharp, E-commerce in your inbox: Product recommendations at scale, in: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 1809–1818.
- [4] M. Li, S. Zhou, D. Wang, G. Chen, Identifying influential nodes based on resistance distance, *J. Comput. Sci.* 67 (2023) 101972.
- [5] S. Sevgican, M. Turan, K. Gökarslan, H.B. Yilmaz, T. Tugcu, Intelligent network data analytics function in 5G cellular networks using machine learning, *J. Commun. Netw.* 22 (3) (2020) 269–280.
- [6] S. Peng, J. Nie, X. Shu, Z. Ruan, L. Wang, Y. Sheng, Q. Xuan, A multi-view framework for BGP anomaly detection via graph attention network, *Comput. Netw.* 214 (2022) 109129.
- [7] Y. Li, S. Xie, Z. Wan, H. Lv, H. Song, Z. Lv, Graph-powered learning methods in the internet of things: A survey, *Mach. Learn. Appl.* 11 (2023) 100441, <http://dx.doi.org/10.1016/j.mlwa.2022.100441>, URL: <https://www.sciencedirect.com/science/article/pii/S2666827022001165>.
- [8] S. Tiwari, F.N. Al-Aswadi, D. Gaurav, Recent trends in knowledge graphs: theory and practice, *Soft Comput.* 25 (2021) 8337–8355.
- [9] Z. Li, F. Liu, W. Yang, S. Peng, J. Zhou, A survey of convolutional neural networks: analysis, applications, and prospects, *IEEE Trans. Neural Netw. Learn. Syst.* 33 (12) (2021) 6999–7019.
- [10] I.D. Mienye, T.G. Swart, G. Obaido, Recurrent neural networks: A comprehensive review of architectures, variants, and applications, *Information* 15 (9) (2024) 517.
- [11] S. Smys, J.I.Z. Chen, S. Shakya, Survey on neural network architectures with deep learning, *J. Soft Comput. Parad. (JSCP)* 2 (03) (2020) 186–194.
- [12] M.M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, P. Vandergheynst, Geometric deep learning: going beyond euclidean data, *IEEE Signal Process. Mag.* 34 (4) (2017) 18–42.
- [13] C. Gao, Y. Zheng, N. Li, Y. Li, Y. Qin, J. Piao, Y. Quan, J. Chang, D. Jin, X. He, et al., A survey of graph neural networks for recommender systems: Challenges, methods, and directions, *ACM Trans. Recomm. Syst.* 1 (1) (2023) 1–51.
- [14] D. He, C. Liang, H. Liu, M. Wen, P. Jiao, Z. Feng, Block modeling-guided graph convolutional neural networks, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, (4) 2022, pp. 4022–4029.
- [15] J.-Y. Ryu, E. Elala, J.-K.K. Rhee, Quantum graph neural network models for materials search, *Materials* 16 (12) (2023) 4300.
- [16] J. Choi, S. Oh, J. Kim, A tutorial on quantum graph recurrent neural network (QGRNN), in: *2021 International Conference on Information Networking, ICOIN, IEEE*, 2021, pp. 46–49.
- [17] V.N. Ioannidis, A.G. Marques, G.B. Giannakis, A recurrent graph neural network for multi-relational data, in: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, 2019, pp. 8157–8161, <http://dx.doi.org/10.1109/ICASSP.2019.8682836>.
- [18] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, 2013, arXiv preprint [arXiv:1312.6203](https://arxiv.org/abs/1312.6203).

- [19] Z. Hu, J. Li, Z. Pan, S. Zhou, L. Yang, C. Ding, O. Khan, T. Geng, W. Jiang, On the design of quantum graph convolutional neural network in the nisq-era and beyond, in: 2022 IEEE 40th International Conference on Computer Design, ICCD, IEEE, 2022, pp. 290–297.
- [20] J. Zheng, Q. Gao, Y. Lü, Quantum graph convolutional neural networks, in: 2021 40th Chinese Control Conference, CCC, IEEE, 2021, pp. 6335–6340.
- [21] J. Preskill, Quantum computing in the NISQ era and beyond, *Quantum* 2 (2018) 79, <http://dx.doi.org/10.22331/q-2018-08-06-79>.
- [22] S.C. Marshall, C. Gyurik, V. Dunjko, High dimensional quantum machine learning with small quantum computers, *Quantum* 7 (2023) 1078, <http://dx.doi.org/10.22331/q-2023-08-09-1078>.
- [23] G. De Luca, A survey of NISQ era hybrid quantum-classical machine learning research, *J. Artif. Intell. Technol.* 2 (1) (2022) 9–15.
- [24] J. Liu, K.H. Lim, K.L. Wood, W. Huang, C. Guo, H.-L. Huang, Hybrid quantum-classical convolutional neural networks, *Sci. China Phys. Mech. Astron.* 64 (9) (2021) 290311.
- [25] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S.Y. Philip, A comprehensive survey on graph neural networks, *IEEE Trans. Neural Netw. Learn. Syst.* 32 (1) (2020) 4–24.
- [26] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, 2016, arXiv preprint [arXiv:1609.02907](https://arxiv.org/abs/1609.02907).
- [27] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, *Adv. Neural Inf. Process. Syst.* 30 (2017).
- [28] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, et al., Graph attention networks, *Stat* 1050 (20) (2017) 10–48550.
- [29] A. Kumar, S.S. Singh, K. Singh, B. Biswas, Link prediction techniques, applications, and performance: A survey, *Phys. A* 553 (2020) 124289, <http://dx.doi.org/10.1016/j.physa.2020.124289>, URL: <https://www.sciencedirect.com/science/article/pii/S0378437120300856>.
- [30] H. Isah, D. Neagu, P. Trundle, Bipartite network model for inferring hidden ties in crime data, in: Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015, ASONAM '15, Association for Computing Machinery, New York, NY, USA, 2015, pp. 994–1001, <http://dx.doi.org/10.1145/2808797.2808842>.
- [31] L. Backstrom, J. Leskovec, Supervised random walks: predicting and recommending links in social networks, in: Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, WSDM '11, Association for Computing Machinery, New York, NY, USA, 2011, pp. 635–644, <http://dx.doi.org/10.1145/1935826.1935914>.
- [32] B. Perozzi, R. Al-Rfou, S. Skiena, DeepWalk: online learning of social representations, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, ACM, 2014, pp. 701–710, <http://dx.doi.org/10.1145/2623330.2623732>.
- [33] T.H. Haveliwala, Topic-sensitive pagerank, in: Proceedings of the 11th International Conference on World Wide Web, 2002, pp. 517–526.
- [34] A. Grover, J. Leskovec, Node2Vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, ACM, New York, NY, USA, 2016, pp. 855–864, <http://dx.doi.org/10.1145/2939672.2939754>.
- [35] F. Mueller, Link and edge weight prediction in air transport networks — An RNN approach, *Phys. A* 613 (2023) 128490, <http://dx.doi.org/10.1016/j.physa.2023.128490>, URL: <https://www.sciencedirect.com/science/article/pii/S0378437123000456>.
- [36] H. Fan, F. Zhang, Y. Wei, Z. Li, C. Zou, Y. Gao, Q. Dai, Heterogeneous hypergraph variational autoencoder for link prediction, *IEEE Trans. Pattern Anal. Mach. Intell.* 44 (8) (2022) 4125–4138, <http://dx.doi.org/10.1109/TPAMI.2021.3059313>.
- [37] G. Zare, N. Jafari Navimipour, M. Hosseinzadeh, A. Sahafi, Network link prediction via deep learning method: A comparative analysis with traditional methods, *Eng. Sci. Technol. Int. J.* 56 (2024) 101782, <http://dx.doi.org/10.1016/j.jestch.2024.101782>, URL: <https://www.sciencedirect.com/science/article/pii/S221509862400168X>.
- [38] J. Chen, J. Zhang, X. Xu, C. Fu, D. Zhang, Q. Zhang, Q. Xuan, E-LSTM-D: A deep learning framework for dynamic network link prediction, *IEEE Trans. Syst. Man, Cybern.: Syst.* 51 (6) (2021) 3699–3712, <http://dx.doi.org/10.1109/TSMC.2019.2932913>.
- [39] F. Xia, K. Sun, S. Yu, A. Aziz, L. Wan, S. Pan, H. Liu, Graph learning: A survey, *IEEE Trans. Artif. Intell.* 2 (2) (2021) 109–127, <http://dx.doi.org/10.1109/TAI.2021.3076021>.
- [40] R. Chauhan, K.K. Ghanshala, R. Joshi, Convolutional neural network (CNN) for image detection and recognition, in: 2018 First International Conference on Secure Cyber Computing and Communication, ICSCCC, 2018, pp. 278–282, <http://dx.doi.org/10.1109/ICSCCC.2018.8703316>.
- [41] L. Wu, L. Noels, Recurrent neural networks (RNNs) with dimensionality reduction and break down in computational mechanics; application to multi-scale localization step, *Comput. Methods Appl. Mech. Engrg.* 390 (2022) 114476.
- [42] A. Sherstinsky, Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network, *Phys. D: Nonlinear Phenom.* 404 (2020) 132306, <http://dx.doi.org/10.1016/j.physd.2019.132306>, URL: <https://www.sciencedirect.com/science/article/pii/S0167278919305974>.
- [43] S.J. Rani, S. Vagdevi, Auto-encoder-based technique for effective detection of frauds in social networks, *Int. J. Inf. Comput. Secur.* 18 (3–4) (2022) 348–364.
- [44] D. Jin, Z. Yu, P. Jiao, S. Pan, D. He, J. Wu, P.S. Yu, W. Zhang, A survey of community detection approaches: From statistical modeling to deep learning, 2021, [arXiv:2101.01669](https://arxiv.org/abs/2101.01669).
- [45] S. Solanki, M. Kumar, R. Kumar, A survey on information diffusion and competitive influence maximization in social networks, *Soc. Netw. Anal. Min.* 15 (2025) <http://dx.doi.org/10.1007/s13278-025-01459-2>.
- [46] M. Hasan, M. Zaki, A survey of link prediction in social networks, *Soc. Netw. Data Anal.* (2011) 243–275, http://dx.doi.org/10.1007/978-1-4419-8462-3_9.
- [47] F. Hu, Y. Zhu, S. Wu, W. Huang, L. Wang, T. Tan, GraphAIR: Graph representation learning with neighborhood aggregation and interaction, *Pattern Recognit.* 112 (2021) 107745, <http://dx.doi.org/10.1016/j.patcog.2020.107745>.
- [48] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, *IEEE Trans. Neural Netw.* 20 (1) (2009) 61–80, <http://dx.doi.org/10.1109/TNN.2008.2005605>.
- [49] C. Gao, J. Zhu, F. Zhang, Z. Wang, X. Li, A novel representation learning for dynamic graphs based on graph convolutional networks, *IEEE Trans. Cybern.* 53 (6) (2022) 3599–3612.
- [50] T. Zhao, G. Liu, D. Wang, W. Yu, M. Jiang, Learning from counterfactual links for link prediction, in: International Conference on Machine Learning, PMLR, 2022, pp. 26911–26926.
- [51] D. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, R.P. Adams, Convolutional networks on graphs for learning molecular fingerprints, 2015, [arXiv:1509.09292](https://arxiv.org/abs/1509.09292).
- [52] R. LaPierre, Springer Nature, 2021.
- [53] J. Van Gael, The role of interference and entanglement in quantum computing, *Semant. Sch.* (2005).
- [54] B. Ogur, I. Yilmaz, The effect of superposition and entanglement on hybrid quantum machine learning for weather forecasting, *Quantum Inf. Comput.* 23 (3&4) (2023) 181–194.
- [55] E. Peters, J. Caldeira, A. Ho, S. Leichenauer, M. Mohseni, H. Neven, P. Spentzouris, D. Strain, G.N. Perdue, Machine learning of high dimensional data on a noisy quantum processor, *Npj Quantum Inf.* 7 (1) (2021) 161.
- [56] V. Havlíček, A.D. Córcoles, K. Temme, A.W. Harrow, A. Kandala, J.M. Chow, J.M. Gambetta, Supervised learning with quantum-enhanced feature spaces, *Nature* 567 (7747) (2019) 209–212.
- [57] F. De Falco, A. Ceschini, A. Sebastianelli, B. Le Saux, M. Panella, Quantum latent diffusion models, *Quantum Mach. Intell.* 6 (2) (2024) 85.
- [58] S.Y.-C. Chen, C.-H.H. Yang, J. Qi, P.-Y. Chen, X. Ma, H.-S. Goan, Variational quantum circuits for deep reinforcement learning, 2020, [arXiv:1907.00397](https://arxiv.org/abs/1907.00397).
- [59] Z. Wang, Z. Liang, S. Zhou, C. Ding, Y. Shi, W. Jiang, Exploration of quantum neural architecture by mixing quantum neuron designs, 2021, [arXiv:2109.03806](https://arxiv.org/abs/2109.03806).
- [60] N. Innan, A. Sawaika, A. Dhor, S. Dutta, S. Thota, H. Gokal, N. Patel, M. Khan, I. Theodonis, M. Bennai, Financial fraud detection using quantum graph neural networks, 2023, <http://dx.doi.org/10.48550/arXiv.2309.01127>.

- [61] N. Singh, M. Kumar, B. Biswas, PQCLP: Parameterized quantum circuit based link prediction in dynamic networks, *Comput. Netw.* 241 (2024) 110210, <http://dx.doi.org/10.1016/j.comnet.2024.110210>, URL: <https://www.sciencedirect.com/science/article/pii/S1389128624000422>.
- [62] M. Kumar, N. Singh, B. Biswas, PQKELP: Projected quantum kernel embedding based link prediction in dynamic networks, *Expert Syst. Appl.* 265 (2025) 125944, <http://dx.doi.org/10.1016/j.eswa.2024.125944>, URL: <https://www.sciencedirect.com/science/article/pii/S0957417424028112>.
- [63] W. Wang, QGHNN: A quantum graph Hamiltonian neural network, 2025, [arXiv:2501.07986](https://arxiv.org/abs/2501.07986).
- [64] B.D.M. Jones, G.O. O'Brien, D.R. White, E.T. Campbell, J.A. Clark, Optimising trotter-suzuki decompositions for quantum simulation using evolutionary strategies, 2019, [arXiv:1904.01336](https://arxiv.org/abs/1904.01336).
- [65] G. Verdon, T. McCourt, E. Luzhnic, V. Singh, S. Leichenauer, J. Hidary, Quantum graph neural networks, 2019, [arXiv:1909.12264](https://arxiv.org/abs/1909.12264).
- [66] H.J. Kimble, The quantum internet, *Nature* 453 (7198) (2008) 1023–1030, <http://dx.doi.org/10.1038/nature07127>.
- [67] K. Qian, Z. Eldredge, W. Ge, G. Pagano, C. Monroe, J.V. Porto, A.V. Gorshkov, Heisenberg-scaling measurement protocol for analytic functions with quantum sensor networks, *Phys. Rev. A* 100 (2019) <http://dx.doi.org/10.1103/PhysRevA.100.042304>, URL: <https://arxiv.org/abs/1901.09042>.
- [68] Z. Qu, X. Liu, M. Zheng, Temporal-spatial quantum graph convolutional neural network based on Schrödinger approach for traffic congestion prediction, *IEEE Trans. Intell. Transp. Syst.* PP (2022) 1–10, <http://dx.doi.org/10.1109/TITS.2022.3203791>.
- [69] J. Zheng, Q. Gao, M. Ogorzalek, J. Lu, Y. Deng, A quantum spatial graph convolutional neural network model on quantum circuits, *IEEE Trans. Neural Netw. Learn. Syst.* PP (2024) <http://dx.doi.org/10.1109/TNNLS.2024.3382174>.
- [70] J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl, Neural message passing for quantum chemistry, in: *Proceedings of the 34th International Conference on Machine Learning, ICML, 2017*, pp. 1263–1272, URL: <https://arxiv.org/abs/1704.01212>.
- [71] K. Xu, W. Hu, J. Leskovec, S. Jegelka, How powerful are graph neural networks? in: *International Conference on Learning Representations, ICLR, 2019*, URL: <https://arxiv.org/abs/1810.00826>.
- [72] A. Micheli, Neural network for graphs: A contextual constructive approach, *IEEE Trans. Neural Netw.* 20 (3) (2009) 498–511.
- [73] M. Niepert, M. Ahmed, K. Kutzkov, Learning convolutional neural networks for graphs, in: M.F. Balcan, K.Q. Weinberger (Eds.), *Proceedings of the 33rd International Conference on Machine Learning*, in: *Proceedings of Machine Learning Research*, 48, PMLR, New York, New York, USA, 2016, pp. 2014–2023, URL: <https://proceedings.mlr.press/v48/niepert16.html>.
- [74] S. Abu-El-Haija, B. Perozzi, R. Al-Rfou, A.A. Alemi, Watch your step: Learning node embeddings via graph attention, *Adv. Neural Inf. Process. Syst.* 31 (2018).
- [75] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, P.S. Yu, Heterogeneous graph attention network, in: *The World Wide Web Conference, 2019*, pp. 2022–2032.
- [76] L. Yu, G. Chen, H. Chen, J. Lv, L. Yang, Enhancing graph learning with quantum structural encoding, *Neurocomputing* (2025) 132054.
- [77] M. Vogel, Quantum computation and quantum information, by M.A. Nielsen and I.L. Chuang, *Contemp. Phys.* 52 (6) (2011) 604–605, <http://dx.doi.org/10.1080/00107514.2011.587535>, <https://doi.org/10.1080/00107514.2011.587535>.
- [78] M. Schuld, A. Bocharov, K.M. Svore, N. Wiebe, Circuit-centric quantum classifiers, *Phys. Rev. A* 101 (2020) 032308, <http://dx.doi.org/10.1103/PhysRevA.101.032308>, URL: <https://link.aps.org/doi/10.1103/PhysRevA.101.032308>.
- [79] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, S. Ahmed, V. Ajith, J.M. Arrazola, A. Atkins, T. Back, L. Banchi, G. Barratt, T. Bromley, A. Delorme, A. Di Paolo, J. van Herck, T. Isaacsson, K. McKiernan, Z. Niu, O. di Matteo, A. Dusko, K.K. Sabapathy, C. Sparaciari, A. Száva, B. Vigoureaux, R. Wendin, M. Zajac, T. Zohren, N. Killoran, PennyLane: Automatic differentiation of hybrid quantum-classical computations, 2018, URL: <https://pennylane.ai>, (Accessed 15 February 2024).
- [80] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, J. Leskovec, Open graph benchmark: Datasets for machine learning on graphs, *Adv. Neural Inf. Process. Syst.* 33 (2020) 22118–22133.
- [81] T. Häner, D.S. Steiger, 0.5 petabyte simulation of a 45-qubit quantum circuit, in: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, ACM, 2017*, <http://dx.doi.org/10.1145/3126908.3126947>.
- [82] Z. Yang, W.W. Cohen, R. Salakhutdinov, Revisiting semi-supervised learning with graph embeddings, 2016, [arXiv:1603.08861](https://arxiv.org/abs/1603.08861).
- [83] M. Zitnik, J. Leskovec, Predicting multicellular function through multi-layer tissue networks, *Bioinformatics* 33 (14) (2017) i190–i198, <http://dx.doi.org/10.1093/bioinformatics/btx252>.
- [84] P. Mernyei, C. Cangea, Wiki-CS: A wikipedia-based benchmark for graph neural networks, 2022, [arXiv:2007.02901](https://arxiv.org/abs/2007.02901).
- [85] J. Leskovec, A. Krevl, SNAP Datasets: Stanford large network dataset collection, 2014, <http://snap.stanford.edu/data>.
- [86] Z. Holmes, N. Coble, A.T. Sornborger, Y. Subaşı, On nonlinear transformations in quantum computation, 2023, [arXiv:2112.12307](https://arxiv.org/abs/2112.12307).
- [87] S. Sim, P.D. Johnson, A. Aspuru-Guzik, Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms, *Adv. Quantum Technol.* 2 (12) (2019) <http://dx.doi.org/10.1002/qute.201900070>.
- [88] A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster, J.I. Latorre, Data re-uploading for a universal quantum classifier, *Quantum* 4 (2020) 226, <http://dx.doi.org/10.22331/q-2020-02-06-226>.
- [89] M. Kumar, S. Mishra, B. Biswas, Features fusion based link prediction in dynamic networks, *J. Comput. Sci.* 57 (2022) 101493.
- [90] M. Zhang, Y. Chen, Link prediction based on graph neural networks, in: *Advances in Neural Information Processing Systems*, 2018, pp. 5165–5175.
- [91] J. Haferkamp, P. Faist, N.B. Kothakonda, J. Eisert, N. Yunger Halpern, Linear growth of quantum circuit complexity, *Nat. Phys.* 18 (5) (2022) 528–532.



Nisha Singh Received her Master of Science in Computer Science from Banaras Hindu University, Varanasi (Uttar Pradesh) in 2008 and Bachelor of Science in CS from Mithibai College, University of Mumbai, Mumbai (Maharashtra) in 2005. She is currently perusing her Ph.D. in Computer Science and Engineering and working as a System Analyst cum Op. In-charge of Supercomputing Centre, Indian Institute of Technology (BHU), Varanasi. Her research interests include Quantum Machine Learning, Link Prediction, Community Detection, and Knowledge Graph completion.



Mukesh Kumar received a Ph.D. in Computer Science and Engineering from the Indian Institute of Technology (BHU), Varanasi, and his Master of Engineering in Computer Networking from the University Visvesvaraya College of Engineering (UVCE), Bangalore (Karnataka) in 2013. He received a Bachelor of Engineering in Information Science & Engineering from Bapuji Institute of Engineering and Technology, Davangere (Karnataka) in 2007. He is currently working as an Associate Professor at Kalinga Institute of Industrial Technology (KIIT), DU, Bhubaneswar. His research interests include Machine Learning, Quantum Computing, Link Prediction, Community Detection, and Influence Maximization in social/complex networks.



Jaideep Sharma is pursuing an Integrated Dual Degree in Computer Science and Engineering from the Indian Institute of Technology (BHU), Varanasi. He is an upcoming Software Engineer at Microsoft India. His research interests include Quantum Computing, Machine Learning, Deep Learning, Development and Algorithms.



Bhaskar Biswas received a Ph.D. in Computer Science and Engineering from the Indian Institute of Technology (BHU), Varanasi. He received the B.Tech. degree in Computer Science and Engineering from Birla Institute of Technology, Mesra. He is working as an Associate Professor at the Indian Institute of Technology (BHU), Varanasi in the Computer Science and Engineering department. His research interests include Data Mining, Text Analysis, Machine Learning, Influence Maximization, Link Prediction, and Social Network Analysis.