CTQW-GraphSAGE: Trainabel Continuous-Time Quantum Walk On Graph

Yangjie XU, Hui Huang, and Radu State

SEDAN, SnT-Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg, Campus Kirchberg, 29 Avenue J.F. Kennedy, L-1855 Luxembourg
{yangjie.xu, huang.hui, radu.state}@uni.lu

Abstract. In recent years, Graph Neural Networks (GNNs) have made significant strides in various applications, demonstrating their potential in handling complex networked data. Simultaneously, quantum machine learning has emerged as a rapidly advancing and promising field, leveraging quantum computing principles to enhance machine learning models. Benefiting from the advancements in both GNNs and quantum machine learning, we propose a novel hybrid model called CTQW-GraphSAGE. This model aims to combine the strengths of classical and quantum approaches to improve performance on graph-related tasks. The model is built on the GraphSAGE framework, enhanced with quantum feature mapping and Continuous-Time Quantum Walk (CTQW). These enhancements are used to calculate aggregation weights for neighboring nodes relative to the target node, thereby integrating quantum properties into the classical model. We evaluate the proposed model on various benchmark datasets and compared our results with several baseline graph neural network methods. CTQW-GraphSAGE achieves comparable results to the classical models on most of the selected datasets on node classification tasks.

Keywords: Graph Neural Networks, Continuous-Time Quantum Walk, Node Sampling.

1 Introduction

Graph-structured data analysis has become an indispensable tool in modern data science, especially Graph Neural Networks (GNNs) has achieved giant success across various domains, including social networks, biological networks, knowledge graphs, and more [1–3]. GNNs are designed to perform computation on graphs, enabling the model to learn and generalize from topological structures. The fundamental operation in GNNs is the message passing or neighborhood aggregation mechanism. In this framework, nodes aggregate feature information from their neighbors, allowing the model to capture both local structures and global graph properties. This iterative process involves updating the representation of each node by aggregating representations of its neighbors and potentially its features. Then, undertake different downstream tasks fit to different domains and scenarios.

However, the traditional neighborhood aggregation strategy only considers the neighboring nodes of the target node, ignoring the more complex topological information. Moreover, the aggregation methods based on pooling and mean represent a fixed-parameters approach, involving no trainable parameters during the aggregation.

To preserve the complex topological information based on target nodes during graph analysis, this paper introduces a k-hop sampling method. This approach extends beyond merely considering a node's immediate neighbors to include structural information with k-hops in the graph. A weighted aggregation that leverages both the topological information and the node features of the graph is then employed to enhance the adaptability and flexibility of the aggregation strategy. The weights for the aggregation are derived using Continuous-Time Quantum Walk (CTQW) within the graph. Furthermore, a variational quantum circuit is utilized to generate parametric embedding, thereby transforming it into a trainable Continuous-Time Quantum Walk.

This paper introduces a quantum-classical hybrid model with an aggregation method based on Continuous-Time Quantum Walk, the so-called CTQW-GraphSAGE. This method aims to aggregate more complex topological information between nodes, not just relying on direct neighbors, and utilizes a parametrically embedded CTQW for weighted information aggregation.

Our main contributions are as follows:

- We utilize a novel approach to graph sampling that emphasizes the hop-based selection of nodes, diverging from traditional methods that primarily consider neighbors based on their layer-wise proximity. This strategy is designed to capture and preserve the intricate local topological structures surrounding a target node, which are often crucial for understanding the node's role and characteristics within the broader network.
- We employ Continuous-Time Quantum Walk (CTQW) on the subgraph to obtain corresponding aggregation probabilities. Instead, we incorporate the similarity of features between different nodes into the Hamiltonian. This inclusion significantly enhances the representation of node feature aggregation.
- A variational Quantum Circuit (VQC) is utilized to create parametric embedding of node features. This strategic implementation allows for the weights, which are generated through the CTQW process within the graph, to be optimizable. It makes the weights not only dynamic but also learnable to enhance the overall efficacy of the information aggregation process in the graph.

The remaining paper is organized as follows. Section 2 presents related works. Section 3 provides the overview of the proposed approach and detailed designs of each component. Section 4 comprehensively evaluates the proposed approach and compares the performance with the state-of-the-art baselines. Finally, section 5 concludes the paper.

2 Related Work

2.1 Graph Neural Network

In recent years, graph neural networks (GNNs) [4] have emerged as a powerful tool for processing and learning from graph-structured data. Previous approaches using MLP-like [5, 6] networks focused primarily on feature information, ignoring the characteristics of the graph structure itself. A significant leap was the development of Graph Convolutional Networks (GCN) [7], which is simplified graph convolutional using efficient layer-wise propagation rules based on adjacency and degree matrices. This was followed by Graph Attention Networks (GAT and GATII) [8, 9] that introduce an attention mechanism, assigning dynamic importance to neighbors. The difference between spectral and spatial approaches in GNNs further broadened the scope of the field. Spectral methods [10, 11] leverage graph Laplacian's eigenvalues for convolution, offering a mathematical approach. On the other hand, GraphSAGE[12] directly aggregates neighbor features, addressing scalability and enabling inductive learning on large graphs. The challenge of heterogeneous and dynamic graphs led to the development of specialized models like Heterogeneous GNNs (HetGNNs) [13] and Dynamic GNNs (DGCNNs) [14], catering to diverse node types and evolving graph structures. The concept of Message Passing Neural Networks (MPNNs) [15] unified various GNN architectures under a single framework, streamlining the understanding of how GNNs update node features. Besides these, DeepWalk [16] utilizes random walks on the graphs and treats each node visit in a walk as a word. For reducing the computational complexity, SGC [17] removes the non-linearities between layers dan replaces multiple layers with a single line classifier.

The applicability of GNNs extends to various domains. For instance, in social network analysis, GNNs are used for community detection and recommendation systems. In bioinformatics, they assist in protein-protein interaction predictions, and in computer vision, GNNs are applied in scene graph generation and object detection.

2.2 Quantum Machine Learning

Simultaneously, Quantum Machine Learning (QML) as an emerging interdisciplinary field at the interaction of quantum computing and machine learning has developed rapidly these years. Numerous machine learning algorithms have been realized into quantum versions [18–20], offering acceleration for classical problems. Moreover, many scholars and researchers have harnessed the advantages of quantum computing and integrated them with traditional neural networks, constructing a variety of hybrid models for different domains.

The field of Quantum Machine Learning (QML) [21] has witnessed significant growth and innovation over the past decade. Variational Quantum Eigensolvers (VQEs) [22] has emerged as the fundamental quantum algorithm. Quantum Support Vector Machines (QSVM) [23] aims to provide quantum speedup in classification tasks as the analogs of classical support vector machines. Quantum algorithms for feature selection, such as Quantum Principle Component Analysis (QPCA) [24] aim to reduce dimensionality and improve the efficiency of classical machine learning algorithms.

The fusion of quantum computing and machine learning techniques has extended to graph data, presenting novel solutions for graph-related problems. Quantum Graph Neural Networks represent a significant development, combining the power of Quantum Computing and Graph Neural Networks (GNNs). QGNNs aim to perform graph-based machine learning tasks, such as node classification, link prediction, and graph classification, with quantum-enhanced capabilities. Notably, Verdon et al. [25] pioneered the development of the Quantum Graph Neural Network (QGNN), a quantum computing-based model for graph classification. QGNN leverages a quadratic Hamiltonian to encode graph structures and employs quantum circuits to extract pertinent structural details. Though Peter et al. introduced the Equivalent Quantum Graph Circuit (EQGC) [26]. EQGC excels at capturing permutation-invariant topologies of input graphs. Its scalability is constrained, as the number of required qubits scales linearly with the number of nodes, restricting its applicability to small-scale synthetic datasets. A notable breakthrough in quantum graph algorithms is the Graph Quantum Neural Tangent Kernel (GraphQNTK) [27], which stands as the sole quantum algorithm capable of handling realistically sized graph data. For utilizing the advancement of quantum computing, ego-graph based Quantum Graph Neural Network (egoQGNN) [28] greatly reduced the trainable parameters compared the traditional GNNs.

In summary, within the domain of combining Graph Neural Networks (GNNs) and Quantum Machine Learning (QML), numerous networks have achieved success. However, many of these networks have not effectively harnessed the power of Quantum Machine Learning to simultaneously leverage both feature information and graph structure. This article introduces a novel approach that utilizes Quantum Machine Learning for feature mapping and employs Quantum Random Walks for probability computation. This innovative approach has demonstrated excellent results in node classification tasks, showcasing the potential of quantum-enhanced techniques in graph-based machine learning.

3 Methodology

3.1 Preliminaries

A graph G can be described using two matrices: an adjacency matrix A, where $A \in \{0,1\}^{n \times n}$, and a node feature matrix X, where $X \in \mathbb{R}^{n \times d}$. In these matrices, n represents the total number of nodes, and d signifies the dimensionality of the node features. Within the adjacency matrix A, an entry $A[i,j] = 1$ indicates the presence of an edge between nodes v_i and v_j; conversely, $A[i,j] = 0$ implies the absence of an edge. Based on the definition of a graph, some additional notations and explanations that are utilized in subsequent processes are presented in Table 1.

Table 1: Notations and Functions

Symbols	Definition or Meanings
k	The maximum sampling hop
m	The number of k-hop samples for the target node
$N_k^m(v)$	A sampled nodes set of target node v
X_v	Features sets of $N_k^m(v)$
ρ_v	The feature mapping after parametric quantum embedding.
P_v	Probablities of N_v after quantum walk
$\odot$	Hadamard Product (element-wise product)

3.2 Framework

Before the theoretical foundation of the method, this subsection provides an overall introduction to the entire network structure of the method we propose. The architecture of the system is illustrated in Fig. 1.

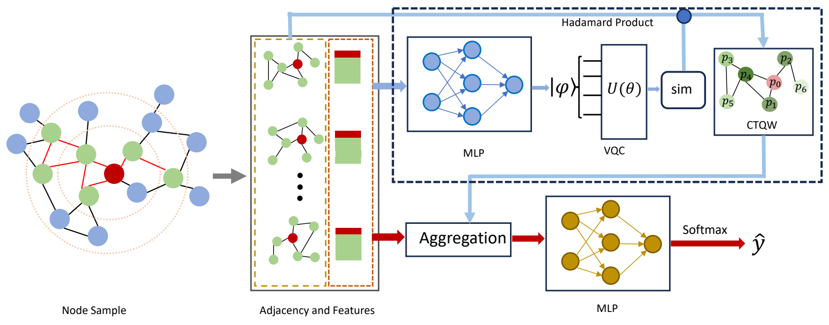


Fig. 1: The overall architecture of CTQW-GraphSAGE.

The whole process depicted begins with node sampling to extract a subgraph, from which the adjacency and feature matrices are obtained. The features undergo a hybrid quantum-classical transformation. The classical component consists of a multilayer perceptron (MLP), which functions to reduce dimensionality. In parallel, the quantum component comprises a variational quantum circuit (VQC), which encodes classical data into a quantum state and performs a quantum feature mapping through the circuit.

The mapped features are then used to calculate similarity measures, which are combined with the sampled adjacency matrix through a Hadamard product. This product result serves as a Hamiltonian operator to govern the time evolution in a continuous-time quantum walk (CTQW), resulting in a probability

Algorithm 1 k-hop Nodes Sampling

Input: Graph: $\{G = (V, E), |V| = N\}$; Node Features $X = \{x_v | \forall v \in V\}$, sample nodes: m; sample hops: k

Output: Adjacency Matrix A_v and Features Matrix X_v of subgraphs.

```
1: Initialize node set $N_k^m(v)$
2: for $i$ in range $(1, k)$ do
3:     Find the $i$-hop neighbours of node $v$ in $G$, add these nodes to the temporary set $T(v)$
4: end for
5: Select $m$ nodes from $T(v)$ and set them as $N_k^m(v)$
6: Generate adjacency matrix $A_v$ and features matrix $X_v$ for $N_k^m(v)$
7: return $N_k^m(v)$, $A_v$, $X_v$
```

distribution. The probabilities represent the expectation of the walker landing on each node within the subgraph.

Leveraging this probability distribution, feature aggregation is performed for the target node. The process concludes with a classical MLP followed by a softmax classifier, which collectively serves to classify the nodes based on the aggregated features. The methodology shows a blend of classical machine learning techniques and quantum computing principles to enhance the analysis of graph-structured data.

3.3 Detail Description

In this subsection, we focus on the critical aspects of the network, including node sampling, the implementation of quantum mapping embedding with VQC, the Continuous-Time Quantum Walk operating on the graph, and feature aggregation.

Sampling: Unlike the layer-wise random sampling in GraphSAGE, where the number of neighbors for each layer and each target node is fixed to ensure consistency and controllability at each layer, the proposed sampling method involves sampling within different hops. The sampling process also retains neighbor information among nodes other than the target node, trying to preserve local structures and complexities. The corresponding adjacency matrix of these nodes and their features are obtained through this sampling, as shown in the Algorithm 1.

The objective of sampling is to obtain a set of nodes from the Graph $G(V, E)$ which consists of vertices V and edges E for each target node v in the vertex set V, the k-hop neighborhood is defined as follows:

$$N_k^m(v) = \{u \mid d(v, u) \leq k\} \tag{1}$$

where $d(v, u)$ is the shortest path distance between nodes v and u. From this k-hop neighborhood, a subset of $N_k^m(v)$ m nodes is randomly selected.

The proposed sample algorithm implies that $N_k^m(v)$ includes the target node v, its immediate neighbors, and all other nodes that can be reached by traversing

Algorithm 2 Trainable Continuous-Time Quantum Walk

```
Input: Adjacency matrices $A_v$, Features matrices $X_v$ and evolution time $t$
Parameter: Parameters of Variable Quantum Circuit and MLP $\theta$, $\omega_q$.
Output: Probabilities
 1: Initialize $\theta$, $\omega_p$ and set $t$
 2: for each node $v$ in $V$ do
 3:     Initialize the quantum state $\psi_0$ by setting the target node $v$ state as $|1\rangle$ and
        other nodes $u$ states as $|0\rangle$
 4:     Features dimensions reduction by MLP: $X_v \mapsto X_v^{'}$
 5:     Encode features to quantum state $X_v^{'} \mapsto |X_v^{'}\rangle$
 6:     Feed the re-encoded features to VQC $|X_v^{'}\rangle \mapsto |\rho_v\rangle$
 7:     Measure and get the output $|\rho_v\rangle \mapsto \rho_v$
 8:     Calculate the similarity matrix $S_v = sim(\rho_v)$
 9:     Continuous-Time Quantum Walk evolution: $|\psi_t\rangle = e^{-iS_v \odot A_v t} |\psi_0\rangle$
10:     Calculate the probabilities vector $P_v$
11: end for
12: return all $P_v$
```

up to k edges. The k-hop sampling strategy is valuable for capturing a comprehensive local context around each node, as it not only considers the direct connections (1-hop neighbors) but also includes nodes that are indirectly connected within k steps. After the sampling, we treat the corresponding adjacency matrices A_v and features set X_v as the input of Trainable Continuous-Time Quantum Walk which includes a variable quantum circuit (VQC) and an evolution operator.

Trainable Continuous-Time Quantum Walk: The Trainable Continuous-Time Quantum Walk (T-CTQW) is composed of two parts. One is a hybrid quantum-classical part which consists of an MLP and a variational quantum circuit. The other one is a Continuous-Time Quantum Walk (CTQW). The overall illustration of a Trainable Continuous-Time Quantum Walk is presented in Algorithm 2.

For each node v in the graph node set V, the quantum state $|\psi_0\rangle$ is prepared in a way that the target node v is set $|1\rangle$, and all other nodes u are set to $|0\rangle$. Feature reduction is then performed on the feature matrices X_v using an MLP, effectively compressing the data before it is encoded into a quantum state. The reduced features $X_v^{'}$ undergo amplitude encoding, transforming them into a quantum state $|X_v^{'}\rangle$. These quantum-encoded features are fed into the VQC, which processes them and generates an output quantum state mapping ρ_v, and it is illustrated in Fig. 2.

For the specialized VQC, which is constructed using a combination of specific quantum gates. These gates include Rx(Pauli-X rotation gate), Ry (Pauli-Y rotation gate), and Rz (Pauli-Z rotation gate), which are each responsible for rotating a qubit around their respective axes on the Bloch sphere. Additionally, the circuit incorporates CNOT (Controlled NOT) gates, which are utilized for

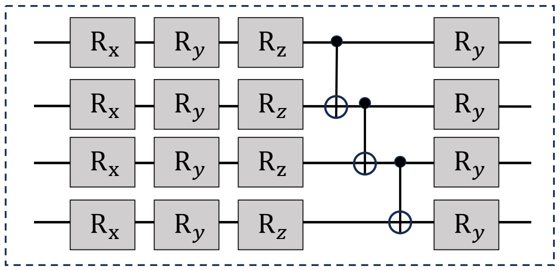


Fig. 2: The VQC architecture.

creating entanglements between qubits, a fundamental characteristic of quantum computing.

Each qubit initially passes through a sequence of parameterized rotation gates and then proceeds through an entanglements section composed of several Controlled-Not gate (CNOT), followed by an additional layer of rotation gates to enhance expressivity. The VQC is represented as $U(\theta)$. Then the mapping process can be described as $|\rho_v\rangle = U(\theta)\,|\rho_v\rangle$. After the measurement $\rho_v = \langle\rho_v|\,M\,|\rho_v\rangle$ where M represent the observation computation.

In a traditional CTQW, the walker's evolution is governed by the system's Hamiltonian $\mathcal{H}$, which is typically related to the adjacency matrix or Laplacian matrix of the graph or lattice on which the walk is taking place. The state of the walker $|\psi\rangle_t$ at time t is given by the solution to the Schrödinger equation:

$$i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle = \mathcal{H}|\psi(t)\rangle \tag{2}$$

Where the Hamiltonian is the adjacent matrix. Solving the Schrödinger equation with $\hbar = 1$, we get the state of the quantum walker at time t:

$$|\psi_t\rangle = e^{-i\mathcal{H}t}\,|\psi_0\rangle \tag{3}$$

This work considers more than just the topological structure of the graph. We aim to provide a richer, higher-dimensional representation that encompasses both the graph's topological information and the connections based on node features. Therefore, the Hamiltonian operator is revised to be the Hadamard product of the adjacency matrix and the feature similarity computed by the output of the VQC.

$$\mathcal{H} = sim(\rho_v) \odot A_v. \tag{4}$$

Equation 4 shows that the Hamiltonian matrix is constructed by first calculating the similarities between specific features mapping output by VQC. Then, we mask these similarity matrices using the adjacency matrices to eliminate expressions of similarity between nodes that are not connected.

To determine the probability of finding the quantum walker at a particular node u which is one of the nodes in the subgraph based on the target node v at time t, we project the time-evolved state onto the basis state $|u\rangle$ corresponding to the node. The probability $P_u(t)$ is calculated as the absolute square of the amplitude:

$$P_u(t) = |\langle u\,|\psi(t)\rangle\,|^2 \tag{5}$$

This process is repeated for each node in the subgraph to obtain the complete probability distribution across all nodes P_v.

In summary, the process of CTQW translates the classical information of a graph, including the connectivities between nodes and their features, into weights used for the aggregation of graph features. This transformation effectively captures the structure and attributes of the graph within a quantum framework, facilitating computations that leverage quantum properties such as superposition and entanglement. The specific mathematical expression of this process can be detailed as follows:

In practical terms, the sum of the probabilities for all nodes should be equal to 1,

$$\sum_u P_u(t) = 1, u \in N_k^m(v), \tag{6}$$

reflecting the normalization condition and ensuring that the quantum walk describes a complete and physically possible quantum system.

This rule is not only a fundamental principle of quantum mechanics but also a key condition that must be adhered to when designing and interpreting quantum algorithms in quantum computing. In practical quantum computing applications, ensuring the normalization of quantum states is crucial, as it guarantees the correctness and reliability of quantum computational results.

Feature Aggregation: In the feature aggregation part, we leverage the probabilities derived from continuous quantum random walks to present a novel approach to synthesizing enriched node representations in graphs. This aggregation method takes into account not only the direct connections of nodes or the topological structure of the graph but also fully integrates the interconnections among features

$$\begin{gathered} h_v^l = \sigma(W(\cdot\{h_v^{l-1}\}\cup\{WeightedMean(h_u^{l-1}, P_u)\})) \\ (\forall u \in N_k^m(u), u \neq v) \end{gathered} \tag{7}$$

The equation describes the process of updating the hidden state of node v at layer l in the neural network. The hidden state h_v^l is computed as a function of the node's previous layer state h_v^{l-1} and the weighted features of its neighboring nodes. The activation function σ introduces nonlinearity. W represents the weight matrix for the linear transformation of node features. The sum is taken over all neighboring nodes u of v, where each neighbor's contribution to the new feature representation is weighted by the probability P_u, derived from processes like quantum random walks. This formulation captures both the direct connections of the nodes and the topological structure of the graph, integrating the relationships among features for a richer representation. In addressing the node classification task, after the described feature aggregation process, we implement a conventional post-processing methodology. This involves the use of a multilayer perceptron integrated with a softmax function to categorize the nodes.

4 Experiments

This section presents the evaluation results of the proposed CTQW-GraphSAGE on various datasets and compares them against the state-of-the-art baselines.

4.1 Datasets

Six datasets, Cora, PubMed, Citeseer, Wisconsin, Texas, and Cornell, are chosen to evaluate the proposed approach. Table 2 contains detailed statistical information about these datasets, and a brief introduction about each dataset is provided below.

Table 2: Datasets Details

Dataset	# Nodes	# Edges	# Features	# Labels
Cora	2708	5429	1433	7
Pubmed	19717	44388	500	3
Citeseer	3327	4732	3703	6
Wisconsin	251	499	1703	5
Texas	183	309	1703	5
Cornell	183	295	1703	5

Cora: This is a citation network dataset where nodes represent scientific papers, and edges represent citations between these papers. The papers are classified into different classes based on their topics, such as Machine Learning, Neural Networks, etc. Each paper (node) is described by a binary word vector indicating the absence/presence of the corresponding word from the dictionary.

PubMed: Similar to Cora, the PubMed dataset is also a citation network but focuses on diabetes-related publications from the PubMed database. Each paper in the dataset is described by a TF/IDF weighted word vector from a dictionary. The papers are categorized into three classes based on the type of diabetes they discuss.

Citeseer: This is another citation network where nodes represent publications and edges represent citations. Like Cora, each document is associated with a word vector indicating the presence of words from a dictionary. The papers are categorized into different classes based on their research topics.

Wisconsin, Texas, and Cornell: These three datasets are part of the WebKB dataset collection and consist of web pages from computer science departments of various universities, categorized into classes like faculty, student, project, etc. Each webpage is described by a word vector similar to the Cora and Citeseer datasets. The datasets represent a network where edges are hyperlinks between these web pages.

Table 3: Performance for the node classification task on six datasets (mean accuracy(%) and standard deviation over 10 trials), In each dataset, the best performance is highlighted in **bold**, while the second-best is underlined.

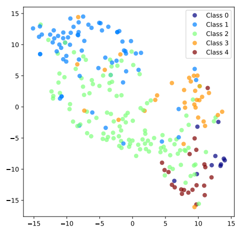
Methods	Cora	Citeseer	Pumbed	Cornell	Wisconsin	Texas
DeepWalk	$82.32_{\pm 0.72}$	$60.78_{\pm 1.11}$	$61.25_{\pm 1.30}$	$50.81_{\pm 5.37}$	$50.39_{\pm 5.48}$	$47.57_{\pm 3.01}$
Graph-based Method (2-layers)						
GCN	$87.39_{\pm 0.27}$	$76.5_{\pm 0.42}$	$87.24_{\pm 0.25}$	$58.65_{\pm 2.1}$	$44.22_{\pm 1.2}$	$44.59_{\pm 1.36}$
GAT	$87.08_{\pm 0.48}$	$77.54_{\pm 0.40}$	$87.28_{\pm 0.28}$	$59.73_{\pm 1.89}$	$46.86_{\pm 3.33}$	$50.27_{\pm 2.16}$
GraphSAGE-mean	$\underline{87.50_{\pm 0.31}}$	$77.21_{\pm 0.42}$	$\underline{87.45_{\pm 0.41}}$	$\underline{75.96_{\pm 3.5}}$	$\underline{72.94_{\pm 1.92}}$	$\mathbf{71.81_{\pm 2.36}}$
GraphSAGE-pool	$86.71_{\pm 0.55}$	$75.60_{\pm 0.54}$	$86.68_{\pm 0.38}$	$71.89_{\pm 2.76}$	$69.22_{\pm 3.40}$	$68.91_{\pm 2.18}$
CTQW-GraphSAGE	$\mathbf{87.89_{\pm 0.95}}$	$\mathbf{78.04_{\pm 0.28}}$	$\mathbf{88.60_{\pm 2.58}}$	$73.64_{\pm 3.08}$	$\mathbf{74.35_{\pm 2.40}}$	$\underline{70.84_{\pm 3.85}}$
Graph-based Method (1-layer)						
SGC	$84.20_{\pm 0.27}$	77.04 ± 0.17	$85.20_{\pm 0.45}$	$60.00_{\pm 1.08}$	$45.29_{\pm 0.59}$	$51.89_{\pm 1.08}$
GCN	$85.45_{\pm 0.52}$	$77.32_{\pm 0.28}$	$86.64_{\pm 0.40}$	$61.08_{\pm 1.79}$	$45.29_{\pm 0.59}$	$49.46_{\pm 1.73}$
GAT	$84.23_{\pm 0.72}$	$77.66_{\pm 0.31}$	$85.18_{\pm 0.49}$	$63.51_{\pm 3.25}$	$47.84_{\pm 3.19}$	$53.24_{\pm 2.72}$
GraphSAGE-mean	$83.93_{\pm 0.26}$	$77.30_{\pm 0.18}$	$84.89_{\pm 0.31}$	$\mathbf{77.29_{\pm 2.16}}$	$72.75_{\pm 1.84}$	$70.54_{\pm 0.81}$
GraphSAGE-pool	$83.02_{\pm 0.40}$	$\underline{77.74_{\pm 0.26}}$	$84.15_{\pm 0.49}$	$73.24_{\pm 2.25}$	$65.50_{\pm 1.80}$	$67.57_{\pm 2.09}$
CTQW-GraphSAGE	$85.47_{\pm 0.35}$	$76.78_{\pm 0.90}$	$86.93_{\pm 0.86}$	$73.47_{\pm 3.40}$	$69.68_{\pm 2.81}$	$69.78_{\pm 2.12}$

4.2 Baselines

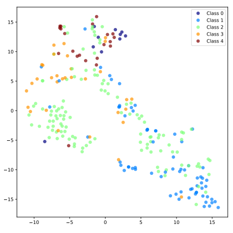
To evaluate the performance of our model on different datasets, we implement several state-of-the-art models, including Simplifying Graph Convolutional Networks (SGC), DeepWalk, Graph Convolutional Networks (GCN), Graph Attention Networks (GAT), and GraphSAGE with pooling and mean aggregation, and compare their performance on the above datasets with our proposed approach.

4.3 Experiments set-up

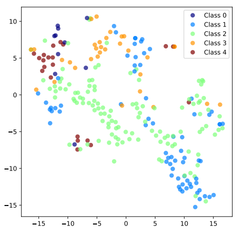
We assess and compare our model based on the accuracy achieved in node classification tasks. To ensure fairness and credibility, we conduct ten trials for each model on every dataset and the accuracy report is based on the average accuracy. For each trial, we run the model 100 epochs. For the neural networks method, we use Adam optimizer. The setting of the hyperparameters: dropout at 0.6, learning rate at 0.001, weight decay = $5e^{-4}$, the hidden dimension within {32, 64, 128}. Especially for the DeepWalk method, the number of walks is 10, the window size is within 5, the embedding size is 128 and the walk length is



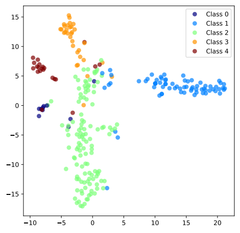
(a) SGC



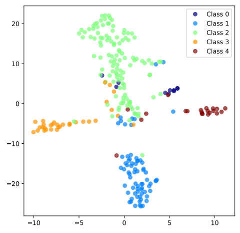
(b) GAT



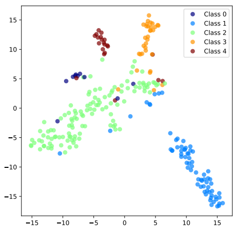
(c) GCN



(d) GraphSAGE-Mean



(e) GraphSAFE-Pool



(f) CTQW

Fig. 3: t-SNE of various networks on the Wisconsin dataset, with subfigures from (a) to (f) representing SGC, GAT, GCN, GraphSAGE with mean aggregation, GraphSAGE with pool aggregation, and our CTQW-GraphSAGE, respectively (2-layer versions for graph-based models).

80. For the proposed method, the number of node samples in the first layer is set to {20, 15}, and in the second layer, it is {10, 5}. The implementation environment includes python (3.8), torch (2.0.1), torch-geometric (2.4)[29], and Paddle-Quantum (2.4.0)[30]. In our study, all experiments are conducted on a Unix device equipped with a Radeon Pro 5500M GPU. We conducted comparative experiments for different neural network architectures by implementing both 1-layer and 2-layer variants. This approach allows us to evaluate the impact of network depth on the model's performance.

4.4 Performance

Table 3 comprehensively demonstrates the performance of various networks across multiple datasets, where we implement experiments with both 1-layer and 2-layer configurations for graph-based networks. Notably, our method achieved the best results in most of the selected datasets, such as Cora, Citeseer, Pubmed, and Wisconsin, and the second-best in the Texas dataset, while performing at an average level on the Cornell dataset compared to other graph-based methods. Both 1-layer and 2-layer networks effectively handled node classification tasks. In terms of stability, compared to DeepWalk, GCN, GAT, and SGC methods, our approach and GraphSAGE adapted well to different datasets for node classification. For instance, as shown in Fig. 3, the t-SNE visualization of the final embeddings of trained networks on the Wisconsin dataset, SGC, GAT, and GCN does not classify the dataset effectively, whereas CTQW and GraphSAGE distinctly differentiated between different nodes.

5 Conclusion

In this work, we introduce a novel hybrid network called CTQW-GraphSAG. This network utilizes a parameterized variational quantum circuit for node feature mapping, followed by computing the similarity between these mappings. We then construct a Hamiltonian using the Hadamard product of the similarity matrix and the adjacency matrix to facilitate the evolution of continuous quantum walks. The resulting probabilities of the walker residing on the nodes are used for weighted aggregation. This approach not only leverages the structural information of the graph but also considers the node feature information in the quantum space. Additionally, in our node sampling process, we take into account the k-hop neighbors of the target node, focusing on preserving the complex information about the target node. This method has also achieved comparable performance to current mainstream graph neural networks in node classification tasks. However, as the quantum part of our method is simulated on classical computers using quantum platforms, it incurs a relatively high time cost for training. In the future, we hope to utilize real quantum devices and more efficient computational methods for acceleration.

References

1. Y. Li, Y. Ji, S. Li, S. He, Y. Cao, Y. Liu, H. Liu, X. Li, J. Shi, and Y. Yang, "Relevance-aware anomalous users detection in social network via graph neural network," in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021, pp. 1–8.
2. H. Zhang, Z. Feng, and C. Wu, "A non-local graph neural network for identification of essential proteins," in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022, pp. 1–8.
3. C. Ren, L. Zhang, L. Fang, T. Xu, Z. Wang, S. Yuan, and E. Chen, "Ontological concept structure aware knowledge transfer for inductive knowledge graph embedding," in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021, pp. 1–8.
4. J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI open*, vol. 1, pp. 57–81, 2020.
5. H. Taud and J. Mas, "Multilayer perceptron (mlp)," *Geomatic approaches for modeling land change scenarios*, pp. 451–455, 2018.
6. Y. Hu, H. You, Z. Wang, Z. Wang, E. Zhou, and Y. Gao, "Graph-mlp: Node classification without message passing in graph," *arXiv preprint arXiv:2106.04051*, 2021.

7. T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.
8. P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.
9. S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks?" *arXiv preprint arXiv:2105.14491*, 2021.
10. Y. Ma, J. Hao, Y. Yang, H. Li, J. Jin, and G. Chen, "Spectral-based graph convolutional network for directed graphs," *arXiv preprint arXiv:1907.08990*, 2019.
11. A. Qin, Z. Shang, J. Tian, Y. Wang, T. Zhang, and Y. Y. Tang, "Spectral–spatial graph convolutional networks for semisupervised hyperspectral image classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 16, no. 2, pp. 241–245, 2018.
12. W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.
13. C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous graph neural network," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 793–803.
14. F. Manessi, A. Rozza, and M. Manzo, "Dynamic graph convolutional networks," *Pattern Recognition*, vol. 97, p. 107000, 2020.
15. J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Message passing neural networks," *Machine learning meets quantum physics*, pp. 199–214, 2020.
16. B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701–710.
17. F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *International conference on machine learning.* PMLR, 2019, pp. 6861–6871.
18. D. Dong, C. Chen, H. Li, and T.-J. Tarn, "Quantum reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 5, pp. 1207–1220, 2008.
19. P.-L. Dallaire-Demers and N. Killoran, "Quantum generative adversarial networks," *Physical Review A*, vol. 98, no. 1, p. 012324, 2018.
20. M. Henderson, S. Shakya, S. Pradhan, and T. Cook, "Quanvolutional neural networks: powering image recognition with quantum circuits," *Quantum Machine Intelligence*, vol. 2, no. 1, p. 2, 2020.
21. M. Schuld, I. Sinayskiy, and F. Petruccione, "An introduction to quantum machine learning," *Contemporary Physics*, vol. 56, no. 2, pp. 172–185, 2015.
22. J. Tilly, H. Chen, S. Cao, D. Picozzi, K. Setia, Y. Li, E. Grant, L. Wossnig, I. Rungger, G. H. Booth *et al.*, "The variational quantum eigensolver: a review of methods and best practices," *Physics Reports*, vol. 986, pp. 1–128, 2022.
23. P. Rebentrost, M. Mohseni, and S. Lloyd, "Quantum support vector machine for big data classification," *Physical review letters*, vol. 113, no. 13, p. 130503, 2014.
24. S. Lloyd, M. Mohseni, and P. Rebentrost, "Quantum principal component analysis," *Nature Physics*, vol. 10, no. 9, pp. 631–633, 2014.
25. G. Verdon, T. McCourt, E. Luzhnica, V. Singh, S. Leichenauer, and J. Hidary, "Quantum graph neural networks," *arXiv preprint arXiv:1909.12264*, 2019.
26. P. Mernyei, K. Meichanetzidis, and I. I. Ceylan, "Equivariant quantum graph circuits," in *International Conference on Machine Learning.* PMLR, 2022, pp. 15 401–15 420.

27. Y. Tang and J. Yan, "Graphqntk: Quantum neural tangent kernel for graph data," *Advances in Neural Information Processing Systems*, vol. 35, pp. 6104–6118, 2022.
28. X. Ai, Z. Zhang, L. Sun, J. Yan, and E. Hancock, "Decompositional quantum graph neural network," *arXiv preprint arXiv:2201.05158*, 2022.
29. M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.
30. "Paddle quantum," 2020. [Online]. Available: https://github.com/PaddlePaddle/Quantum