On the Design of Quantum Graph Convolutional Neural Network in the NISQ-Era and Beyond

Zhirui Hu[1], Jinyang Li[1], Zhenyu Pan[2], Shanglin Zhou[3]
Lei Yang[1], Caiwen Ding[3], Omer Khan[3], Tong Geng[2], Weiwen Jiang[1]

[1]George Mason University [2]University of Rochester [3]University of Connecticut
{zhu2, jli56, lyang29, wjiang8}@gmu.edu
zpan21@ur.rochester.edu; tong.geng@rochester.edu; {shanglin.zhou, caiwen.ding, omer.khan}@uconn.edu

***Abstract*—The rapid growth in the size of Graph Convolutional Neural Networks (GCNs) encounters both computational- and memory-wall on classical computing platforms (e.g., CPU, GPU, FPGA, etc.). Quantum computing, on the other hand, provides extremely high parallelism for computation. Although quantum neural networks have been recently studied, the research on quantum graph neural networks is still in its infancy. The key challenge here is how to integrate both the graph topology information and the learning ability of GCNs into quantum circuits. In this work, we leverage the Givens rotations and its quantum implementation to encode graph information; in addition, we employ the widely used variational quantum circuit to bring the learnable parameters. On top of these, we present a full-quantum design of Graph Convolutional Neural Networks, namely "QuGCN", for semi-supervised learning on graph-structured data. Experiment results show our design is competitive with classical GCNs in terms of node classification accuracy on Cora sub-dataset. More importantly, we show the potential advantages that can be achieved by the proposed quantum GCN design when the number of features grows.**

***Index Terms*—Graph Convolutional Neural Network, Quantum circuit design, Givens rotation, NISQ.**

I. Introduction

Due to the consistently growing size of machine learning models and the high parallelism of the quantum computing paradigm, Quantum Machine Learning (QML) has become one of the most active emerging topics. The basic concept of QML is to perform machine learning tasks on quantum devices, such as quantum feedforward neural network [1]–[3], quantum convolutional neural network [4], and quantum recurrent neural network [5]. Recently, Graph Convolutional Neural Networks (GCN), a deep learning method designed to process graph-structured data, raise much attention. GCN can handle the Non-Euclidean datasets that are not suitable for traditional neural networks. Given the graph structure and nodes' information as input, GCNs can work on graph tasks such as node/graph classification and edge prediction [6], e.g., Semi-Supervised Classification with Graph Convolutional Networks [7], [8]. Due to the high complexity of graphs, classical computing platforms (e.g., CPUs, GPUs, and FPGAs) met bottlenecks in both storage and computation. With the ability to represent 2^N features on N qubits, quantum computing has great potential for GCN applications. However, there are limited research efforts on quantum GCNs; in particular, it lacks a design of quantum-version GCN to leverage the power of quantum computing in the near-term Noisy Intermediate-Scale Quantum (NISQ) era.

Although the quantum computing platform has demonstrated its ability to accelerate standard neural networks, (e.g., feedforward neural network, FFNN), when quantum computing meets GCNs, new challenges arise. In the conventional FFNN, it only contains a parameterized classifier (or called weight matrix), which can be easily realized by a Variational Quantum Circuit (VQC); on the other hand, instead of a weight matrix, GCN further has an adjacency matrix to represent the topology of a given graph. Existing quantum GCN works either only implement the weight matrix [9] or use classical computing for the adjacency matrix [10]. The former approach cannot extract the features from the graph structure, while the classical computation in the latter approach can easily become the performance bottleneck. The key challenge here is how to implement the adjacency matrix on the quantum circuit. Second, the design should be scalable, targeting the near-term quantum devices; in particular, the quantum GCN should be accommodated to the limited number of qubits. Existing designs apply $O(N)$ or even more qubits to represent N nodes in GCN, which is obviously not scalable. To overcome the above challenges, innovations are needed in the design of quantum circuits for both adjacency matrix and weight matrix with a limited number of qubits.

In this paper, we propose a brand new design to implement the quantum circuit for GCN, namely quantum graph convolutional neural network (QuGCN). In QuGCN, we apply Givens rotation to realize the message passing with neighbor nodes, which has the same function as the adjacency matrix. Second, we employ amplitude encoding to represent the node features in the quantum circuit, as such, N node features can be encoded to $\log N$ qubits. Last, we seamlessly attach the VQC to the quantum circuit with the graph information; i.e.,

Z. Hu, J. Li, and W. Jiang are with the Department of Electrical and Computer Engineering and Quantum Science and Engineering Center at Mason. L. Yang is with the Department of Information Sciences and Technology at Mason. C. Ding and S. Zhou are with the Department of Computer Science and Engineering at UConn. O. Khan is with the Department of Electrical and Computer Engineering at UConn. Z. Pan and T. Geng are with the Department of Electrical and Computer Engineering at URos.

2576-6996/22/$31.00 ©2022 IEEE
DOI 10.1109/ICCD56317.2022.00050

the circuit implemented by Givens rotation for the adjacency matrix. As such, the proposed design can present the structure feature of a given GNN using a small number of qubits, and perform GCN tasks, such as node or graph classification.

The main contributions of this paper are as follows.

- We propose an end-to-end design, namely QuGCN, to implement graph convolutional neural networks to quantum circuits to process graph-structured data.
- We bring the Givens rotation and variational quantum circuit into the QuGCN design so that the adjacency matrix and weight matrix in the graph neural network can be successfully encoded to the quantum circuits.
- Experiments are conducted on a commonly used dataset to evaluate the effectiveness of QuGCN, on top of which, we provide the insights the design of quantum GCN and point out the future directions on QGCN.

QuGCN is evaluated on the commonly used dataset, Cora. With the comparison of multiple baselines, the proposed QuGCN can outperform the existing quantum GNN without integrating the graph's topology information, in terms of accuracy. The node classification accuracy is similar to the GCN in classical computing. What's more, we analyze the design cost complexity of QuGCN and the classical GNN model. With the increase of input features, the total cost of QuGCN can be exponentially reduced from a classical GCN. In the meanwhile, we also show that for the nodes with 256 features, QuGCN can outperform classical GCN when the number of nodes in the graph is less than 128.

The remainder of the paper is organized as follows. Section II provides the preliminaries and reviews the related work; Section III presents the proposed QuGCN design. The detailed quantum circuit design based on Givens rotation is presented in Section IV. Experimental results are reported in Section V. Section VI discusses the insights of QuGCN and concluding remarks are given in Section VII.

II. Preliminaries and Related Work

A. Quantum Basics

The basic unit in the quantum computing is the quantum bit, called qubit. It is a linear combination of two basis states: $|\phi\rangle = a|0\rangle + b|1\rangle$, where $|0\rangle$ and $|1\rangle$ are the basis states. Coefficients a and b are known as amplitudes, which are complex numbers and satisfy $a^2 + b^2 = 1$. For a n-qubit system, a vector, $\mathbf{x}$ with 2^n complex elements, is used to represent the amplitudes of 2^n basis quantum states. All the elements in $\mathbf{x}$ satisfy $\sum_{i=0}^{2^n} |x_i|^2 = 1$.

A set of qubits is composed of a quantum circuit, and the computation is to transit the qubits from one state to another. Here, the basic computation unit is the quantum gate. A quantum gate can be represented by a unitary matrix, $U(\theta)$, where θ is a trainable parameter. The quantum computation is the transition of qubits' state, e.g., $|\phi\rangle = \cdots U(\theta)|\phi_0\rangle$, where ϕ_0 is the initial state and $|\phi\rangle$ is the output state. In a quantum circuit, a set of quantum gates are performed sequentially to realize a function.

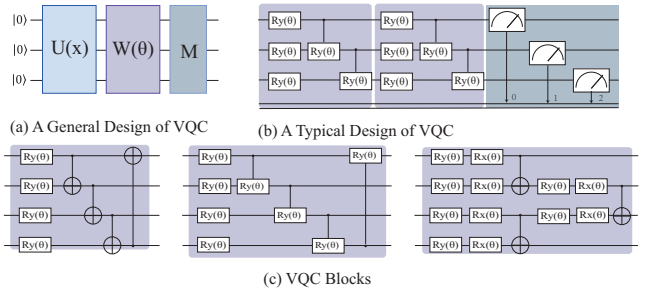

Fig. 1. Background of VQC: (a) A general design of VQC. (b) A typical design of VQC without encoding. The purple block is repeated for certern times. (c) Three commonly-used VQC design with trainable parameters and strong entanglement.

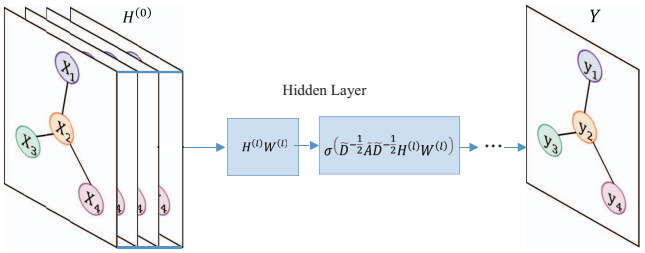

Fig. 2. Background of Graph Convolutional Network (GCN): The high-level depiction of multi-layer GCN for semi-supervised learning

At the end of the quantum circuit, a readout subcircuit is applied to extract the computation results, which measures the probability of a qubit state, say state $|0\rangle$ for Z-measurement.

Variational Quantum Circuit (VQC) [11] is commonly used to perform learning tasks in quantum computing [12]. Figure 1(a) shows the general design of VQC, including encoding unit $U(x)$, computation unit $W(\theta)$, and measurement subcircuit M. For data encoding $U(x)$, there are different ways to convert classical data to quantum data, such as amplitude encoding , angle encoding [13], etc. In computation unit W, the qubits are entangled to represent one function, which include a set of trainable parameters $\boldsymbol{\theta} = [\theta_1, \theta_2, \ldots \theta_n]$. After the measurement sub-circuit M, the results are usually connected with an activation function, which is similar to classical neural networks to perform further tasks (e.g., classification). With these components, Figures 1(b)(c) give the commonly-used VQC designs.

B. Graph Convolutional Neural Networks

Graph Convolutional Neural Network (GCN) [7] shows its potential to process graph tasks (e.g., graph classification and semi-supervised node classification), where the data have graph structures opposite to the data with regular structure, like images. The key idea of GCN is to use edge structure to aggregate node information and generate new node representations with regular structure. As such, the newly generated node representations can be processed using the conventional deep neural networks, such as Multi-Layer Perceptron (MLP). Existing GCN works [7] have shown the remarkable ability of GCN to accomplish fast and scalable semi-supervised classification of nodes for given graphs.

Figure 2 illustrates an example of a multi-layer GCN for semi-supervised learning. The hidden layers include two parts, feature extraction and edge representation, sharing the same graph structure (i.e., edges shown as black lines) for each layer and aggregate node information through the spatial neighborhood. At the end of the last layer, we obtain a set of nodes having the label Y_i, and we will need to predict the labels of the rest nodes in semi-classification tasks. In the above process, one key component is the layer-wise propagation, in the following equation, we give a commonly used rule for such propagation,

$$H^{(l+1)} = \left(H^{(l)}, A\right) = \sigma\left(AH^{(l)}W^{(l)}\right) \tag{1}$$

where A represents the adjacency matrix, W represents the weight matrix, $H^l \in R^{N\times D}$ is the matrix of features in the lth layer; $AH^{(l)}W^{(l)}$ selects the first-order neighbor nodes to realize the information transmission. Equation 1 can successfully complete the information transmission; however, it makes the values of nodes with more neighbors become an infinite number during iterations. This problem can be addressed by normalizing the adjacency matrix as follows,

$$H^{(l+1)} = f\left(H^l, A\right) = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{2}$$

C. Givens Rotation

Givens rotation is originally from the numerical linear algebra, which is a rotation in the plane spanned by two coordinates axes [14]. The fundamental representation of a Givens rotation is as follows:

$$G^N(i,j,\theta) = \begin{bmatrix} 1 & \dots & 0 & \dots & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \dots & c & \dots & s & \dots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \dots & -s & \dots & c & \dots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \dots & 0 & \dots & 1 \end{bmatrix} \tag{3}$$

where $c = \cos\theta$ and $s = \sin\theta$ appear at the i-th and j-th rows and columns. More detailly, given i, j, where i >j, the non-zero elements in the matrix of Givens roation are as follows:

$$g_{kk} = c \quad \text{for } k = i, j$$

$$g_{ji} = -g_{ij} = -s$$

Givens rotation can be employed for computing the QR decomposition of a matrix. One advantage over QR decomposition using Givens rotations is that they can easily be parallelized, and another is that they have a lower operation count for very sparse matrices.

D. Related Work

G Verdon et al. [15] conducted one of the first quantum graph neural network studies. They introduced a general Quantum Graph Neural Network ansatz, a parameterized quantum circuit representing quantum processes with graph structure. To be more specific, they applied the concept of Hamiltonian evolutions to simulate the graph structure. Graphs can naturally describe it if we consider two connected nodes as two qubits with interaction. Based on this proposed ansatz, they gave different specialized architectures.

J Zheng et al. [10] later proposed a quantum Graph Convolutional Neural Networks model aiming to demonstrate the graph's topology in a quantum architecture similar to traditional graph convolutional neural networks. The main task of the paper was to design a quantum circuit that can solve graph-level problems(i.e., graph classification). With the test on the dataset, they could distinguish the input image into two categories.

Limited by the number of usable qubits in the quantum circuit, the previous methods can only handle small graphs. X Ai et al. [16] gave a more comprehensive model to not only achieve the goal of simulating Graph Neural Networks to classify graphs but also propose a strategy to solve the lack of available qubits. The key in their method was to use Subgraph decomposition and CNOT gates to handle the topology of a given graph. For a given graph with n nodes, they split the whole graph into n subgraphs. Thus, each subgraph consisted of a node with its neighbors, represented by qubits equal to the number of subgraph's nodes and followed by a series of trainable parameterized gates. Then, they could entangle their information by applying CNOT gates to each pair of nodes in a subgraph. Finally, they combined all subgraphs and got the graph representation for classification.

Besides the pure quantum circuits, some hybrid methods adopt quantum layers and classical layers to solve machine learning problems. C Tüysüz et al. [9] proposed a Hybrid quantum-classical graph neural network to reconstruct the track of particles(i.e., edge prediction). Their model had three components, the Input network(classical neural network) used to increase the dimension of node features, and the Edge network and the Node network used to update the graph's features. In their design of the Edge network and Node network, they applied trainable classical layers before and after one quantum neural network. The model could predict the connection between two nodes with the extracted edge features in the last Edge network.

The above methods attempt to use the quantum circuits to simulate the graph structure and do neural network tasks. They do not integrate the graph neural networks with quantum machine learning, i.e., encoding the adjacency matrix (representation of a graph) into quantum. Thus, they do not consider the graph's topology in the circuit design. Here, we develop a novel quantum machine learning algorithm for graph-structured data called Quantum Graph Convolutional Neural Network (QuGCN), which implements the quantum circuit design of GCN.

III. Quantum Graph Convolutional Neural Networks (QuGCN)

A. Design Principles

Before introducing the details of the QuGCN design, we will first present our design philosophy.

Principle 1: The quantum version GCN should include all operations in a classical GCN, as shown in Formula 2.

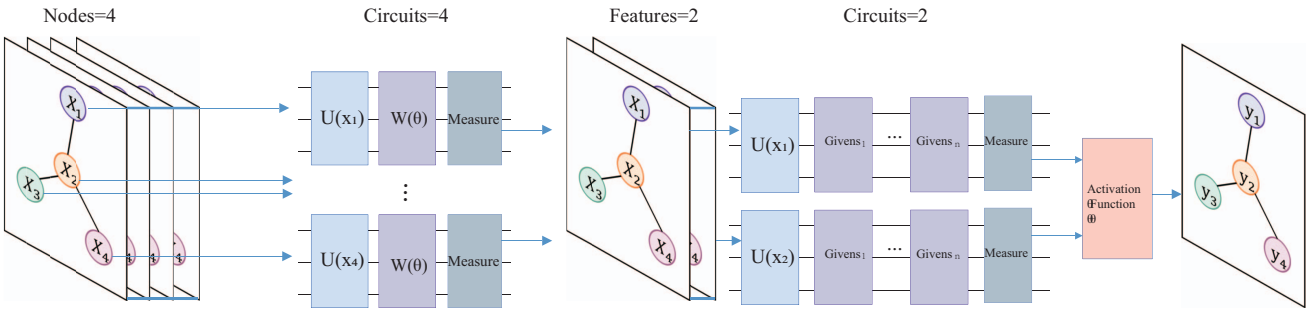

Fig. 3. An overview of Quantum Graph Convolutional Neural Networks (QuGCN)

In our design, QuGCN will perform three operations on a quantum circuit to process the given graph for a learning task: (1) Represent node features in the quantum circuit. (2) Embed graph structure using the quantum operations, called "Graph Structure Embedding". (3) A quantum neural network model to perform learning tasks (e.g., the semi-supervised classification of nodes in a graph), called "Feature Extraction" in this paper.

Principle 2: Design cost can be optimized by leveraging the property that operations (2) and (3) can be swapped. As shown in Formula 2, the order of multiplication between $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}} \times H^{(l)}$ and $H^{(l)} \times W^{(l)}$ will not affect the function. However, if the number of features is decreasing along layers, we can reduce the quantum circuit complexity by firstly executing $H^{(l)} \times W^{(l)}$.

With the assumption that the number of features is decreasing along layers, the proposed QuGCN design is illustrated in Figure 3. It contains two sequential steps, each of which will have a block to encode node features into the quantum circuit, denoted as "Node Representation" in the figure. The first step takes the given graph-structure data as input, and propagates a trained VQC to obtain the intermediate graph-structure data. Then, the second step will process the obtained data according to the graph structure. Finally, the output of the second step will pass an activation function to perform the ML task (e.g., node classification in this figure). In the following of this section, we will introduce these steps in detail.

B. Node and Feature Representation

In this work, we apply amplitude encoding to do the quantum state preparation for both node and feature. For the node representation, our purpose is to map feature vectors $X \in \mathbb{R}^{N\times d}$ to the amplitudes of quantum states, where N is the number of nodes (i.e., vector) and d is the dimensions of a vector (i.e., the number of features in each node). In dimension j, there are N features corresponding to N nodes, denoted as $X_{:j}$, which will be encoded in the j^{th} quantum circuit. To correctly encode N classical features to the quantum circuit, we will firstly normalize $X_{:j}$ by L2-norm to satisfy $\sum_{i=0}^{N} |x_{ij}|^2 = 1$. If $N < 2^n$, where n is the number of qubits, we fill the empty positions with zero. In all, there are d quantum circuits in total. Similarly, for the feature representation, we will encode d features of the i^{th} node in one quantum circuit, and there will be N quantum circuit in total.

C. Feature Extraction Based on VQC

In classical GCN (Equation 2), W is a layer-specific, node-shared, and trainable weight matrix. The shape of the weight matrix will depend on the dimensions of input features and output features in each layer. A simple fully connected (FC) layer is usually used as the trainable weight matrix. To mimic the FC function in GCN, in QuGCN, we employ the variational quantum circuit (VQC) to extract features. With the feature representation introduced in Sec. III-B, we will have N quantum circuits, each of which has the encoded features corresponding to a node. Similar to weight matrix W, all these circuits will propagate the layer-specific and node-shared VQCs.

D. Graph Structure Embedding Based on Givens Rotation

In Equation 2, $\widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}$ represents the graph structure, which realizes the information communication with the first-order neighbor nodes. Similarly, we need to achieve the information transmission with neighbor nodes on quantum circuits. Based on the node representation in Sec. III-B, we will have d quantum circuits with the encoded N features. Since each node corresponds to a quantum state, the communication between nodes in the graph is now equivalent to the information communication between quantum states. That is, we need a quantum gate/circuit to operate on two quantum states at one time and it will not affect other states. Kindly note that since such an operation is performed by the quantum gates, indicating that the operation should be a unitary matrix. To satisfy the above needs, Givens rotation (see Sec II.C) can be used for graph structure embedding.

Givens rotation can intuitively pass the amplitude of the p state to the q state. For a Givens rotation $G^N(p,q,\theta) \in R^{N\times N}$, only the elements of rows p, q and columns p, q are different from the identity matrix $I^N \in R^{N\times N}$. Let $\theta = \frac{\theta}{2}$, the elements of row p, q and column p, q are selected as the submatrices:

$$G^2 = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & \sin\left(\frac{\theta}{2}\right) \\ -\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix} \quad (4)$$

The state transmission process is as $S_{i+1} = G^N(p,q,\theta) \times S_i$, where the state vector $S_i \in \mathbb{R}^{N\times 1}$. The other elements can

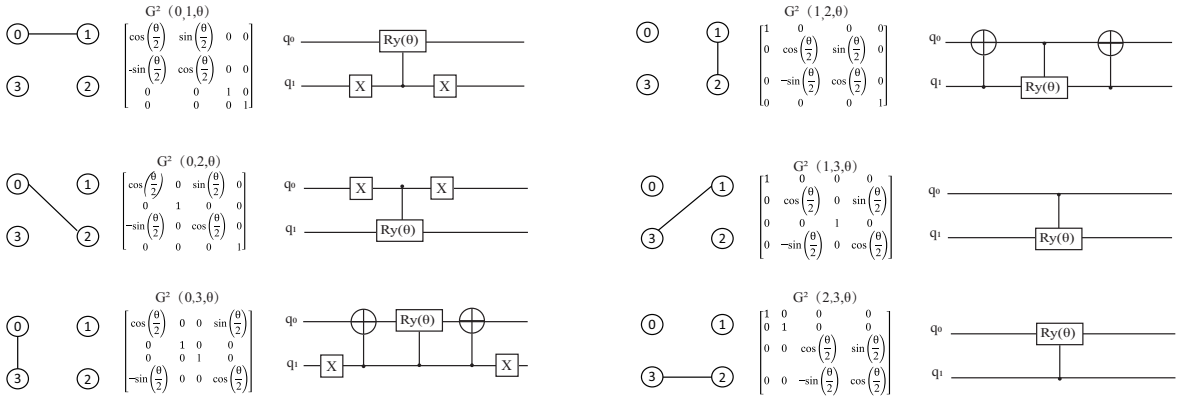

Fig. 4. The quantum implementation of 4-node edge mapping

remain unchanged except for the positions of p and q in the state vector.

Example 1: $N = 2, p = 0, q = 1, s_0 = [a, b]^T$

$$S_1 = G^2(0,1,\theta)\, S_0 = \begin{bmatrix} a \times cos\left(\frac{\theta}{2}\right) + b \times sin\left(\frac{\theta}{2}\right) \\ a \times sin\left(\frac{\theta}{2}\right) + b \times cos\left(\frac{\theta}{2}\right) \end{bmatrix} \quad (5)$$

From Equation 5, it is clear that the involved states correspond to two neighbor nodes. To enable such operation in a N qubits system, the communication between a pair of arbitrary nodes can be formulated as follows,

$$\begin{aligned} G^N|p\rangle &= G^N(p,p,\theta)|p\rangle + G^N(p,q,\theta)|q\rangle \\ G^N|q\rangle &= G^N(q,p,\theta)|p\rangle + G^N(q,q,\theta)|q\rangle \end{aligned} \quad (6)$$

where p, q is the index of matrix G^N. Except $|p\rangle$ and $|q\rangle$, other basis states are left unchanged. In this way, we will generate N Givens rotation to represent the graph structure for an N-nodes graph. Note that θ is a hyperparameter, which is the same in all implemented Givens rotations in QuGCN.

Givens rotations can represent directed graphs. For an undirected graph, we only consider the $givens(p, q, \theta)$ where $p \leq q$.

IV. Circuit Implementation of Givens Rotation

As discussed above, any two quantum states can be communicated by a Givens rotation. For an example of two states, the matrix to represent Givens rotation is Equation 4. In quantum computation, the matrix representation of Ry quantum gate is exactly the same as the matrix of Givens rotation. Therefore, the communication of edge $\langle 0, 1\rangle$ in Example 1 can be implemented by a Ry gate in a quantum circuit. To extend the implementations of quantum circuit for the arbitrary scale of Givens rotation, there are two steps: First, we need a design of 2-qubit quantum circuit for a Givens rotation with the dimension of 4×4. Then, for a quantum system with n qubits, we need to apply the designed 2-qubit quantum circuit for arbitrary 2 states.

A. Two-qubit Quantum Circuit for Givens Rotation

To design the quantum circuit for Givens rotation, we will need to involve the following sets of quantum gates. First, the

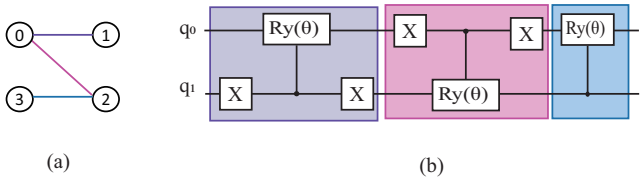

Fig. 5. An example of generating a circuit from a graph.

not gate, X. The matrix representation is as follows:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad (7)$$

Second, the control-not gate, CX. Its matrix representation is as follows:

$$CX = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (8)$$

Lastly, we will invovle the control-Ry gate, whose matrix representation is as follows.

$$CRY = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\left(\frac{\theta}{2}\right) & \sin\left(\frac{\theta}{2}\right) \\ 0 & 0 & -\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix} \quad (9)$$

By combining several CX, X, and CRY gates, we can obtain all the 2-qubits implementations for 4-nodes edge representation. We summarize these implementations in Figure 4. With the design of edge representation for any pair of nodes, we can construct the node representations in a given graph. Figure 5 shows an example of constructing the quantum circuit for a graph.

B. Givens Rotation in a Multi-qubit quantum circuit

Now, we will introduce how to leverage the above 2-qubit circuit design to perform the communication between two states in a N-qubit circuit. Specifically, 2-qubit implementation $G^2(p, q, \theta)$ can be extended to multi-qubit $G^N(p, q)$ by a multi-control unitary Gate. In the following, we will use an example to show the communication between nodes $p = 13$ and $q = 18$ in a 5-qubit circuit. The method can be further extended to the quantum circuit with more qubits.

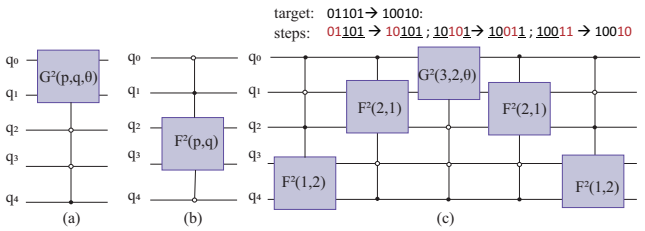

Fig. 6. (a) The implementation of $C_{0100}G^2(p,q,\theta)$ (b) The implementation of $C_{10}F^2(p,q)C_{10}$ (c) The implementation of $G^5\,(13,\ 18)$

TABLE I
COMPLEXITY ANALYSIS. L IS NUMBER OF LAYERS, N IS NUMBER OF NODES, D IS NUMBER OF FEATURES AND E IS NUMBER OF EDGES.

	GCN	QuGCN
HW	$O(LND^2)$	$O(LNlog_2(D))$
AH	$O(L(E+N)D)$	$O(Llog_2^2(N)ED)$
AHW	$O(LND^2+LED)$	$O(LNlog_2(D)+Llog_2^2(N)ED)$

Example 2: the implementation of $G^5\,(13,\ 18)$ can be completed in 3 stages:

In the first stage, we will get a binary number of p and q to represent the node. For example, p = 13: 01101, q = 18: 10010.

Next, in the second stage, we will generate the transmission steps from left to right. Our target is to transmit the amplitude value from the $p-th$ state to the $q-th$ state. In one step, only 2 bits of binary p can be changed by $G^2\,(p,q,\theta)$, and other bits need to remain unchanged. We repeat the two-qubit transmission from left to right by adding 1 to the index i. In order to minimize the steps, if the i-th bit in the binary of p and the binary of q are the same, we could skip the i-th step. For example, we generate the steps for $G^5\,(13,\ 18)$ as follows. Step i=1: 01101 ->10101; Step i=2 (skip): 10101->10011; Step i=3: 10101->10011; Step i=4: 10011 ->10010.

At last, we need to generate the multi-control unitary gates for each step. Multi-controls U-gates with states can achieve only the qubits with U-gates are processed while other qubits remain unchanged, noting $C_{state}UC_{state}$. (Figure 6(a)(b) are examples of $C_{state}UC_{state}$. The last step is with Givens rotations, noting $G^n\,(p,q)$, while all pre-steps are with flip gates, noting $F^n\,(p,\ q)$. Givens rotations are used to do information transmission between 2 states, which is not for arbitrary states. Therefore, we need flip gates to exchange the amplitude between different states. CX gate is an example of flip gate, the matrix representation of which is Equation 8. In a Givens rotation circuit, $F^n\,(p,\ q)$ should be added symmetrically to prevent changing the other states. The circuit is shown in Figure 6(c).

C. Complexity Analysis

We divide the GCN or QuGCN model into two parts to analyze the complexity. The two parts are graph structure embedding (noted as AH) and feature extraction(noted as HW). For simplicity, we assume the number of features is fixed for all layers. At the same time, we dismiss the sparsity of the feature matrix but only focus on the sparsity of the adjacency matrix.

For classical GCN, we do HW first and then do AH. We consider both H and W as dense matrices, so the complexity of HW is summed up to $O(LND^2)$ for L layers. We consider normalized adjacency matrix $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ as a sparse matrix, so the operation number of AH is $L||A||_0F$, where $||A||_0$ is the number of non-zero elements in the normalized adjacency matrix. Considering that normalization will not change the sparsity pattern of the adjacency matrix thoroughly except for diagonal elements, the operation number of $||A||_0$ is $2E+N$ and the complexity of AH is $O(L(E+N)D)$.

For quantum GCN, the execution order of HW and AH is the same as QuGCN. In this paper, we consider the number of basic gates 'U, CU, X, CX, CCX, P' as the operation number of quantum circuits, where C_n is multi-control gate. For commonly-used VQC design, the operation number is rn, where r is a constant value and n is the number of qubits. So the complexity of HW is $O(LNlog_2(D))$. For Givens rotations, the complexity of n-control U-gate $C_{state}UC_{state}$ is $O(n)$ and it should be repeated $O(n)$ times according to step 3, so the complexity of $G_n(p,q)$ or $F_n(p,q)$ is $O(n^2)$ and the complexity of $A\times H$ is $O(LNlog_2(D)+Llog_2^2(N)ED)$.

We summarize the complexity of GCN and QuGCN in Table I. We can clearly learn from the last row that QuGCN can achieve exponential speedup on F but lose the advantage because of the superfluous $O(log_2^2(N))$.

V. EXPERIMENTS

A. Experiment Setups

Datasets We follow the experiment in [7] to evaluate QuGCN on semi-supervised classification tasks in citation networks, Cora. We randomly generate different sub-datasets of Cora with different sizes (ranged from 128 to 1024) to do a binary classification task. Label rate denotes the number of labeled nodes used for training divided by the total number of nodes in each dataset. We do not use the validation set labels for training.

QuGCN models We apply the amplitude encoding presented in [13]. In detail, we convert the input values to the amplitudes by L2-normalization. For the weight matrix, we repeat the main part in VQC 7 times. The classification results are obtained based on the measurement results. We divide the outputs into two groups and sum them up in each group to generate the output values. Finally, we adopt Softmax as the activation function for classification. The QuGCN model is implemented using Qiskit APIs and Torch-Quantum [17], which can be executed on the IBM Qiskit Aer simulator

Competitors We compare QuGCN with the existing works on the same dataset. Our design QuGCN consists of Givens rotations and variational quantum circuits. To verify the effectiveness, the baselines are (1) Classical GCN [7]: normalized adjacency matrix and a fully connected linear layer. (2) Hybrid QGNN [9]: classical normalized adjacency matrix and variational quantum circuits (3) QNN [11]: only variational quantum circuits.

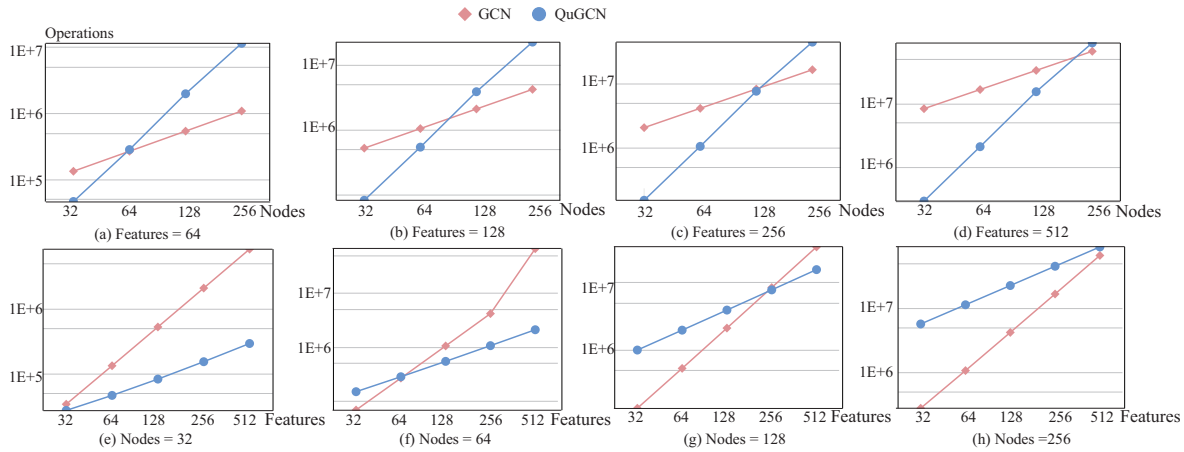

Fig. 7. Comparison between classical GCN and QuGCN on the number of operations: (a-d) with the fixed number of features, operations grows along with the number of nodes; (e-h) with the fixed number of nodes, operations grows along with the number of features.

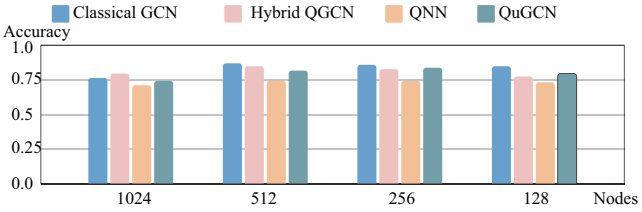

Fig. 8. Comparison of accuracy between QuGCN and other baselines

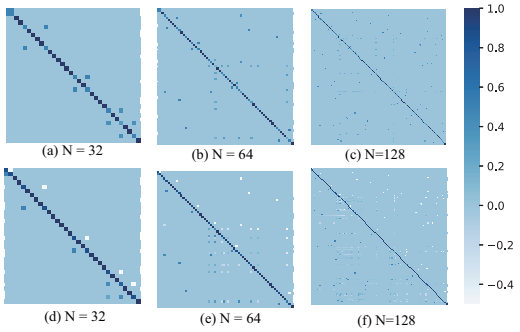

Fig. 9. Visualization of adjacency matrix with heatmap: (a-c) classical GCN 32 to 128 nodes; (d-f) QuGCN with corresponding number of nodes.

B. Design Cost and Accuracy Comparison

Figure 7 shows the increase of operations affected by the number of nodes and features, which verify the conclusion of complexity. In section 4, we have analyzed the complexity of GCN and QuGCN, showing that QuGCN can achieve exponential speedup on F but lose the advantage because of the superfluous $O(log_2^2(N))$. In the experiments, we calculated the operation number on the subset of Cora. When the number of nodes is a relatively small fixed number and the number of features dominates complexity, GCN grows faster than QuGCN. In contrast, when the number of nodes dominates complexity, QuGCN grows faster than GCN. We have to admit that the current implementation of Givens rotations in this paper is not a good design.

Figure 8 reports the accuracy of our proposed model QuGCN and the other four baselines. We learn that the accuracy of QuGCN is higher than QNN on datasets of different sizes. From this result, we can verify that Givens rotations could embed the graph structure to quantum circuits and achieve information transmission between neighbor nodes, which makes the accuracy better in semi-supervised classification tasks. Comparing Hybrid QGNN with classical GCN, the accuracy of the classical GCN layer is usually higher than that of Hybrid QGNN. This is mainly because the number of trainable parameters in VQC is limited due to barren plateaus in quantum neural network training landscapes. Besides, our proposed model can beat the hybrid method in some situations, which shows the effectiveness of Givens rotations to replace the adjacency matrix.

C. Visualization

Figure 9 shows the heatmap of the normalized adjacency matrix (the first row) and the product of Givens rotations (the second row). We found that they have similar patterns. First, the diagonal element from top-left to bottom-right is 1. Second, the position of outstanding points is the same in the corresponding two graphs. We also observe differences in these results. The values of outstanding points on both sides of the diagonal are different in the Givens rotation product but are the same in the normalized adjacency matrix. This is mainly because all of the non-zero elements in the classical normalized adjacent matrix are over zero but the non-zero elements in Givens rotations are skew-symmetric ($A^T = -A$, A is called skew-symmetric).

From the similar patterns, we can conclude that a product of Givens rotations can take the place of a normalized adjacency matrix to transmit information between nodes.

VI. Discussion and Insights

A. Potential Speedup against Classical GCN

Noticing that the implementation in this paper is relatively easy to understand even though it has not been the most effective yet. Since the hyperparameter in each Givens rotation is the same, we can merge the Givens rotations groups in a specific pattern by multiplication. For example,

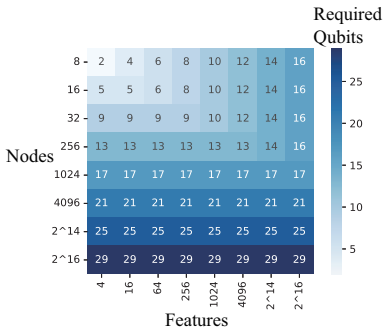

Fig. 10. The heatmap of qubits for growing size of features and nodes

$G_2(0,1) \times G_2(2,3)$ can be implemented by only one RY gate, which makes quantum operations achieve advantages.

Besides, classical GCN is a kind of batch operation, which is suitable for parallel computing. Considering the parallelism of quantum computing, there is enormous potential for QuGCN.

In our future work, we will explore the speedup to show the great potential of quantum implementation of GCN.

B. Scalability on Large-scale Graph

We know that there are currently no stable quantum computers with large-scale qubits; our design shows the scalability on large-scale graphs even in today's small-scale NISQ quantum computers. Figure 10 reports the required number of qubits to implement a QuGCN model with a certain number of nodes and features. As the supplied number of qubits increases, we can map exponential-increase information to quantum qubits.

C. Sparsity of Features and QNN Compression

It should be noticed that the feature vectors of Cora are also sparse. If we consider feature sparsity into computation, the complexity of HW in Equation 2 is $O(LN||W||_0)$, where $||W||_0$ is the number of non-zero elements in weight matrix. $O(LN||W||_0)$ is much better than $O(LND^2)$.

In the meanwhile, feature sparsity can also makes quantum circuit faster. For sparse feature, large-size VQC is not necessary. Existing works [18], [19] show the effectiveness of quantum neural network compression.

VII. Conclusion

In this work, we make the very first design to map the graph convolutional neural networks (GCNs) to the quantum circuit, including the implementation of the weight matrix by the variational quantum circuit and the adjacency matrix. In this way, both graph information and learnable parameters are integrated into the quantum circuit. Experiments are conducted on the subset of the widely used Cora dataset. Results show that the proposed QuGCN design can achieve competitive accuracy with classical GCN. In addition, QuGCN significantly outperforms the existing quantum implementation without integrating the graph topology information. Last, we provide insights on the design to show the potential advantage of quantum computing for GCNs.

Acknowledgement

This work was supported in part by the George Mason University Quantum Science & Engineering Center and the National Science Foundation under Grant No. CNS-1718481.

References

[1] W. Jiang, J. Xiong, and Y. Shi, "A co-design framework of neural networks and quantum circuits towards quantum advantage," *Nature communications*, vol. 12, no. 1, pp. 1–13, 2021.

[2] F. Tacchino, P. Barkoutsos, C. Macchiavello, I. Tavernelli, D. Gerace, and D. Bajoni, "Quantum implementation of an artificial feed-forward neural network," *Quantum Science and Technology*, vol. 5, no. 4, p. 044010, 2020.

[3] W. Jiang, J. Xiong, and Y. Shi, "When machine learning meets quantum computers: A case study," in *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2021, pp. 593–598.

[4] I. Cong, S. Choi, and M. D. Lukin, "Quantum convolutional neural networks," *Nature Physics*, vol. 15, no. 12, pp. 1273–1278, 2019.

[5] J. Bausch, "Recurrent quantum neural networks," *Advances in neural information processing systems*, vol. 33, pp. 1368–1379, 2020.

[6] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. v. d. Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European semantic web conference*. Springer, 2018, pp. 593–607.

[7] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[8] H. Gao, Z. Wang, and S. Ji, "Large-scale learnable graph convolutional networks," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 1416–1424.

[9] C. Tüysüz, C. Rieger, K. Novotny, B. Demirköz, D. Dobos, K. Potamianos, S. Vallecorsa, J.-R. Vlimant, and R. Forster, "Hybrid quantum classical graph neural networks for particle track reconstruction," *Quantum Machine Intelligence*, vol. 3, no. 2, pp. 1–20, 2021.

[10] J. Zheng, Q. Gao, and Y. Lü, "Quantum graph convolutional neural networks," in *2021 40th Chinese Control Conference (CCC)*. IEEE, 2021, pp. 6335–6340.

[11] S. Y.-C. Chen, C.-H. H. Yang, J. Qi, P.-Y. Chen, X. Ma, and H.-S. Goan, "Variational quantum circuits for deep reinforcement learning," *IEEE Access*, vol. 8, pp. 141 007–141 024, 2020.

[12] Z. Wang, Z. Liang, S. Zhou, C. Ding, Y. Shi, and W. Jiang, "Exploration of quantum neural architecture by mixing quantum neuron designs," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2021, pp. 1–7.

[13] R. LaRose and B. Coyle, "Robust data encodings for quantum classifiers," *Physical Review A*, vol. 102, no. 3, p. 032420, 2020.

[14] W. contributors, "Givens rotation — Wikipedia, the free encyclopedia," 2022, online; accessed 18-August-2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Givens rotation&oldid=1095579649

[15] G. Verdon, T. McCourt, E. Luzhnica, V. Singh, S. Leichenauer, and J. Hidary, "Quantum graph neural networks," *arXiv preprint arXiv:1909.12264*, 2019.

[16] X. Ai, Z. Zhang, L. Sun, J. Yan, and E. Hancock, "Decompositional quantum graph neural network," *arXiv preprint arXiv:2201.05158*, 2022.

[17] H. Wang, Y. Ding, J. Gu, Z. Li, Y. Lin, D. Z. Pan, F. T. Chong, and S. Han, "Quantumnas: Noise-adaptive search for robust quantum circuits," in *The 28th IEEE International Symposium on High-Performance Computer Architecture (HPCA-28)*, 2022.

[18] Z. Hu, P. Dong, Z. Wang, Y. Lin, Y. Wang, and W. Jiang, "Quantum neural network compression," *arXiv preprint arXiv:2207.01578*, 2022.

[19] H. Wang, Y. Ding, J. Gu, Y. Lin, D. Z. Pan, F. T. Chong, and S. Han, "Quantumnas: Noise-adaptive search for robust quantum circuits," in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2022, pp. 692–708.