Quantum Walk Neural Networks for Graph-Structured Data

Stefan Dernbach[1], Arman Mohseni-Kabir[1], Siddharth Pal[2], and Don Towsley[1]

[1] University of Massachusetts Amherst, Amherst, MA 01003, USA,
{dernbach, towsley}@cs.umass.edu, arman@physics.umass.edu
[2] Raytheon BBN Technologies, Cambridge, MA 02138,
siddharth.pal@raytheon.com

Abstract. In recent years, neural network architectures designed to operate on graph-structured data have pushed the state-of-the-art in the field. A large set of these architectures utilize a form of classical random walks to diffuse information throughout the graph. We propose quantum walk neural networks (QWNN), a novel graph neural network architecture based on quantum random walks, the quantum parallel to classical random walks. A QWNN learns a quantum walk on a graph to construct a diffusion operator which can then be applied to graph-structured data. We demonstrate the use of this model on a variety of prediction tasks on graphs involving temperature, biological, and molecular datasets.

Keywords: graph neural networks, quantum walks, graph classification, graph regression

1 Introduction

While classical neural network approaches for structured data in the form of images and ordered sequences have been well investigated, there has been growing interest in extending neural network architectures beyond grid-structured data [17] to the more general domain of graph-structured data [4,9,12,13,16,18,19,25]. A subset of these graph deep learning architectures are based on classical random walks to extract and learn information from the graph. We detail several of them here for comparison to our quantum random walk approach.

Diffusion convolution neural networks [4] are a spatial convolutional method that performs random walks on the graph and combines information from spatially close neighbors. DCNNs use powers of the transition matrix $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$ to diffuse information

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053 (the ARL Network Science CTA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on. This document does not contain technology or technical data controlled under either the U.S. International Traffic in Arms Regulations or the U.S. Export Administration Regulations.

across the graph, where $\mathbf{A}$ is the adjacency matrix, and $\mathbf{D}$ is the diagonal degree matrix such that $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$.

Graph convolution networks [16], was proposed to tackle semi-supervised learning using a CNN architecture built via a localized approximation of spectral graph convolutions [9]. The GCN works in a similar fashion as DCNN described above. Here the augmented adjacency matrix $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and degree matrix $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$ replace $\mathbf{A}$ and $\mathbf{D}$. $\tilde{\mathbf{A}}$ can be equivalently thought of as the adjacency matrix of G with a self-loop on every node.

While many graph neural network models are inspired by convolutional networks used for image processing, other classical deep learning models have also seen translations to the graph domain. One such model, graph attention networks [25] are inspired by attention models common in natural language processing and weight importance of different node neighbors. This nonuniform weighting of neighbors is a shared aspect with quantum walk neural networks.

Quantum walks have previously been used in graph kernel methods. Bai et al. [5] propose a kernel method to compare pairs of graphs based on the density matrices of the corresponding quantum walk evolution. Our proposed quantum walk neural network (QWNN) technique is a new graph neural network architecture based on discrete-time quantum walks [1,3,21]. To the best of our knowledge, this paper represents a first step at developing a quantum neural network technique that can operate on graph-structured data. We show that the proposed approach obtains better or competitive results in comparison to other state-of-the-art graph neural network approaches on a wide range of graph datasets, suggesting that quantum techniques should be further investigated in the domain of graph-structured data.

The rest of the paper is organized as follows. The setting of quantum walks on graphs is studied in Section 2, followed by a formal description of the proposed quantum walk neural network technique in Section 3. Experimental results on node and graph regression and graph classification tasks are described in Section 4, followed by a brief discussion on the limitations of our approach in Section 5. Finally, concluding remarks and a discussion on future work is presented in Section 6.

2 Graph Quantum Walks

Quantum walks are a universal model for quantum computation [10]. They offer an alternative approach for implementing a variety of quantum algorithms, including database search [24], graph isomorphism [20], network analysis and navigation, and quantum simulation. Quantum walks are the quantum parallel to their classical counterparts – While a classical walker is modeled by a probability distribution over positions in a graph, a quantum walker is described by a superposition over position states. We utilize the form of a discrete time quantum walk on a general graph as outlined in [2,15]. Given an undirected graph $G = (V, E)$, we define a position Hilbert space $\mathscr{H}_{\mathscr{P}}$ spanned by basis vectors $\left\{\hat{\mathbf{e}}_v^{(p)},\ v \in V\right\}$ and define $\mathscr{H}_{\mathscr{C}}$, the coin space, as an auxiliary Hilbert space of dimension d_{max} spanned by the basis vectors $\left\{\hat{\mathbf{e}}_i^{(c)},\ i \in 1, \ldots, d_{max}\right\}$, where i corresponds to edges incident on a vertex and d_{max} is the maximum degree of the

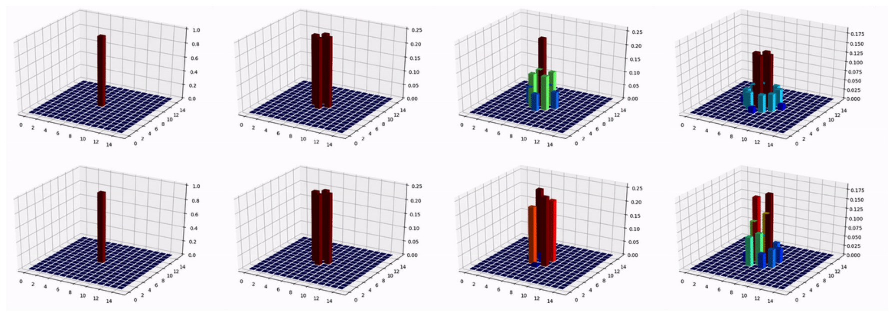
Fig. 1. The probability distribution of a classical random walk (Top) and a quantum random walk (Bottom) on a lattice over four steps (Left to Right). The walks diverge after two steps.

graph. For purpose of brevity, we will use d instead of d_{max}. These states form the spin directions of a walker at vertex v.

A quantum walk has an associated Hilbert space $\mathscr{H}_{\mathscr{W}} = \mathscr{H}_{\mathscr{P}} \bigotimes \mathscr{H}_{\mathscr{C}}$, which is the tensor product of position and coin Hilbert spaces. The basis states in this Hilbert space are of the form $\hat{\mathbf{e}}_v^{(p)} \otimes \hat{\mathbf{e}}_i^{(c)}$, where $v \in V$ spans over set of all vertices, and i is the i^{th} edge incident on node v. Therefore, the position vector of a quantum walker can be written as a linear combination of position state basis vectors, $\sum_{v \in V} \alpha_v \hat{\mathbf{e}}_v^{(p)}$, for some coefficients $\{\alpha_v,\ v \in V\}$ satisfying $\sum_v |\alpha_v|^2 = 1$; while the spin state at each vertex $v \in V$ can also be expressed as linear combinations of spin state basis vectors, $\sum_{i \in 1,\ldots,d} \beta_{v,i} \hat{\mathbf{e}}_i^{(c)}$, for some coefficients $\{\beta_{v,i},\ i \in 1,\ldots,d\}$ satisfying $\sum_i |\beta_{v,i}|^2 = 1$. The coefficients $|\alpha_v|^2$ and $|\beta_{v,i}|^2$ indicate the probabilities of finding the walker at vertex v and the walker having spin i at that vertex respectively when a measurement is done.

A single step in the quantum random walk consists of first applying a coin operator $\mathscr{C}$ that transforms the spin state at a vertex through a unitary transformation acting on the coin Hilbert space $\mathscr{H}_{\mathscr{C}}$. After applying the coin operator, a unitary shift operator $\mathscr{S}$ swaps the states of two vertices connected by an edge, i.e., for an edge (u,v) if u is the i^{th} neighbor of v and v is the j^{th} neighbor of u, then the coefficient corresponding to the basis state $\hat{\mathbf{e}}_v^{(p)} \otimes \hat{\mathbf{e}}_i^{(c)}$ gets swapped with that of the basis state $\hat{\mathbf{e}}_u^{(p)} \otimes \hat{\mathbf{e}}_j^{(c)}$. Let $\Phi^{(t)}$ in $\mathscr{H}_{\mathscr{W}}$ denote the superposition state of the quantum walker at time t. If $\Phi^{(0)}$ is the initial state of the quantum walker on G then after t time steps the state of the walker is described by $\Phi^{(t)}$=$U^t \Phi^{(0)}$, where a single step of the discrete quantum walk on graph G can be expressed in shorthand as $U = \mathscr{S}(\mathscr{C} \otimes I_{|V|})$. Note that the quantum walks behave very differently from classical random walks in that they are heavily influenced by the choice of the initial superposition as well as the coin operator. This allows more degrees of freedom to the deep learning technique to fit the data through a controlled diffusion process. Figure 1 shows how the quantum walk differs from a classical walk over a lattice in that the probability distribution of the classical walk is uniform over space which is not true for the quantum walk.

3 Quantum Walk Neural Networks

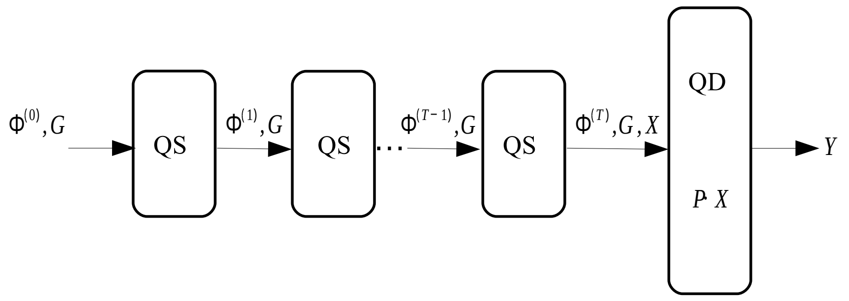

Fig. 2. A schematic diagram of the QWNN architecture. The initial superposition $\Phi^{(0)}$, and graph G, are fed to the quantum step (QS) layers to obtain the final superposition $\Phi^{(T)}$, which in turn is fed to the quantum diffusion (QD) layer along with the feature matrix X to obtain the output Y.

Our proposed neural network architecture, Quantum Walk Neural Network (QWNN), learns a quantum random walk on a graph by means of learning the coin operators and/or the initial superposition of the walker. We further suggest two separate implementations of a QWNN based on whether the coin operators are defined spatially or temporally. In the spatial setting, there is a separate coin operator for each node in the graph. In the temporal case, every node in the graph shares the same coin operator, but that operator is unique to each step in the walk. The first setting allows the quantum walk to behave differently in different sections of the graph while the latter setting allows the walk dynamics to evolve over time.

A QWNN then uses the walker distribution induced by the quantum random walk to form a diffusion operator which act on an input feature matrix. Given a tensor $\boldsymbol{\Phi}^{(0)} \in \mathbb{R}^{W \times N \times d}$ representing a set of W walkers and their corresponding initial superpositions, a set of spatial coins $\mathbf{C} \in \mathbb{R}^{N \times d \times d}$ or temporal coins $\mathbf{C} \in \mathbb{R}^{T \times d \times d}$ and a shift tensor $\mathbf{S} \in \mathbb{Z}_2^{N \times d \times N \times d}$, a quantum walk neural network takes as input a matrix of F features per node $\mathbf{X} \in \mathbb{R}^{N \times F}$, and outputs diffused features $\mathbf{Y} \in \mathbb{R}^{N \times F}$. Note that, when spatially distributing coins, the dimension of the coin on each node equals its degree, but for the sake of implementation we use d universally and set the extra coin variables to zero. For temporally distributed coins, the graph is made d-regular by adding self-loops to nodes.

The network architecture is composed of a sequence of quantum step layers leading to a diffusion layer. Each quantum step layer takes as input the shift tensor which encodes the graph structure and the current superposition tensor. The layer evolves the superposition as follows: For an edge (u,v), with u being the i^{th} neighbor of v and v being the j^{th} neighbor of u, $\boldsymbol{\Phi}_{wuj}^{(t+1)} = \left(\boldsymbol{\Phi}^{(t)}\mathbf{C}\right)_{wvi}$ and $\boldsymbol{\Phi}_{wvi}^{(t+1)} = \left(\boldsymbol{\Phi}^{(t)}\mathbf{C}\right)_{wuj}$. Hence, we set the shift tensor $\mathbf{S}$ as $S_{ujvi} = 1$ iff u is the the i^{th} neighbor of v and v is the j^{th} neighbor of u. The output $\boldsymbol{\Phi}^{(t+1)}$ is fed into either the next quantum step layer or a final diffusion layer. The diffusion layer accepts a superposition tensor $\boldsymbol{\Phi}$ and transforms it into a diffusion matrix **P** by summing the squares of the spin states at each vertex to calculate a transition probability. It then uses **P** to diffuse the input feature matrix **X** across the graph and outputs a new feature matrix **Y**.

For the first layer in the network, we initialize $\boldsymbol{\Phi}^{(0)}$ by constructing a unique walker at each node in the graph with equal spin along each incident edge. If we are learning initial amplitudes, $\boldsymbol{\Phi}^{(0)}$ is stored as a variable that is updated during backpropagation. The method for a forward pass of the complete network (using spatial coins) is given in Algorithm 1. Equation $\boldsymbol{\Phi}^{(t)}_{\cdot i \cdot} \leftarrow \boldsymbol{\Phi}^{(t-1)}_{\cdot i \cdot} \cdot \mathbf{C}_{i \cdot\cdot}$ is replaced with $\boldsymbol{\Phi}^{(t)}_{\cdot i \cdot} \leftarrow \boldsymbol{\Phi}^{(t-1)}_{\cdot i \cdot} \cdot \mathbf{C}_{t \cdot\cdot}$ in the temporal coin case (i.e. coins are indexed spatially rather than temporally). The notation $\mathbf{A} \cdot \mathbf{B}$ denotes the inner product between tensors **A** and **B**, the operator $\mathbf{A} : \mathbf{B}$ is the inner product of **A** and **B** over two dimensions, and $\mathbf{A} \odot \mathbf{B}$ is an elementwise product. The final diffusion equation $\mathbf{Y} \leftarrow \mathbf{PX}$ is similar to other classical graph neural networks that use a random walk diffusion operation. However, there is no explicit weight matrix in the network because it is implicit in learning the random walks.

Algorithm 1: QWNN Forward Pass

```
   given  : Initial Superpositions $\boldsymbol{\Phi}^{(0)}$, Coins $\mathbf{C}$, Shift $\mathbf{S}$
   input  : Features $\mathbf{X}$
   output : Diffused Features $\mathbf{Y}$
1  for $t = 1$ to $T$ do
2      for All nodes $v_i$ do
3          $\boldsymbol{\Phi}^{(t)}_{\cdot i \cdot} \leftarrow \boldsymbol{\Phi}^{(t-1)}_{\cdot i \cdot} \cdot \mathbf{C}_{i \cdot\cdot}$
4      end
5      $\boldsymbol{\Phi}^{(t)} \leftarrow \boldsymbol{\Phi}^{(t)} : \mathbf{S}$ (i.e., $\boldsymbol{\Phi}^{(T)}_{wuj} = \sum_v \sum_i \boldsymbol{\Phi}^{(T)}_{wvi} S_{viuj}$)
6  end
7  $\mathbf{P} \leftarrow \sum_{\mathrm{k}} \boldsymbol{\Phi}^{(T)}_{\cdot\cdot k} \odot \boldsymbol{\Phi}^{(T)}_{\cdot\cdot k}$
8  $\mathbf{Y} \leftarrow h(\mathbf{PX} + \mathbf{b})$
```

Previous work on quantum walks [2] use Grover's diffusion operator as the coin operator because it is the only nontrivial, real-valued transform that is (1) unitary and (2) treats all edges connected to a vertex identically. The first requirement guarantees that superposition of each walker retain unit norm. The second restriction is added in order to prevent edge ordering from affecting the walk. In a QWNN we can choose to relax these conditions for certain tasks to allow for learning biased coin operators. Removing the unitary constraint allows us to learn expansive or contractive diffusion operators. In the spatial coin setting, learning can adapt to any edge ordering equally, so the second condition is unnecessary.

When the unitary property of the coin operator is desirable, we can initialize each coin to any unitary matrix. However, in general, gradient updates to the coin operators cause them to leave the space of unitary matrices. The result of this is that the matrix **P** is no longer a probability matrix and P_{uv} no longer describes the likelihood of a walker beginning on node u and ending on node v of the graph. Several methods have been proposed in the literature to enforce that a unitary matrix receiving gradient updates remains unitary. Within neural networks, the common use case for this is recurrent neural networks. In this work, we use the general parametrization method in [14] to ensure unitarity of the coin operators after each gradient update.

3.1 Edge Ordering

When sharing coin operators across nodes, edge ordering plays an important role in determining the span of possible quantum walks. We propose two heuristic approaches to edge ordering that imposes a shared logic to the coin operation between nodes. First is a global ordering of nodes based on their betweenness centrality. The betweenness centrality [8] of node, v_i is calculated as:

$$g(v_i) = \sum_{j \neq i \neq k} \frac{\sigma_{jk}(v_i)}{\sigma_{jk}}$$

where σ_{jk} is the number of shortest paths from v_j to v_k and $\sigma_{jk}(v_i)$ is the number of shortest paths from v_j to v_k that pass through v_i. This ranks all nodes and an edge connected to a higher ranked node will always precede an edge connected to a lower ranked node. In this setting, a walker moving along a higher ranked edge is moving towards a more central part of the graph compared to a walker moving along a lower ranked edge.

The second approach is a local edge ordering where a similarity score between every pair of adjacent nodes is computed and the edges incident to each node are ordered according to the score of the two nodes it connects. This ordering does not induce a global ranking of nodes, each node has its own ranking of neighbors. This approach allows the quantum walk to distinguish between transitioning to more or less similar nodes in the graph. We use a random walk node similarity measure defined as

$$S(v_i, v_j) = \mathbf{W}_i^k \left(\mathbf{W}_j^k\right)^T$$

where $\mathbf{W}^k$ is a (classical) random walk matrix raised to the kth power.

Neither of these methods guarantees that there will not be a need to break ties between edges. However, in practice, these ties often come from symmetries in the graph and do not have a tangible effect on the global behavior of the quantum walk.

4 Experiments

We demonstrate the effectiveness of QWNNs across three different types of tasks: regression at both the node and the graph level and graph classification. We focus on comparisons with three other graph neural network architectures: diffusion convolution neural networks (DCNN) [4], graph convolution networks (GCN) [16], and graph attention networks (GAT) [25].

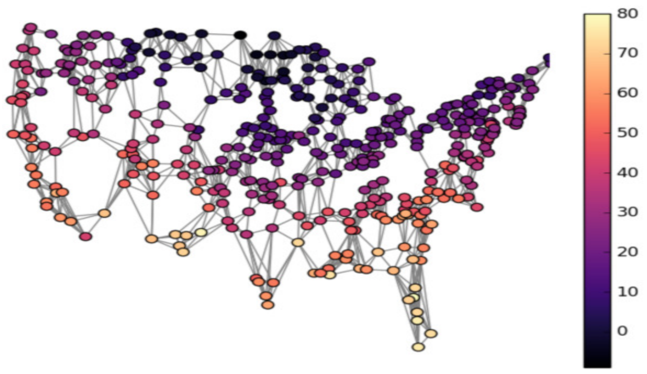

Fig. 3. The nearest neighbor temperature graph with a sample day's temperatures.

4.1 Node Regression

The node regression task is to predict temperatures at 409 locations across the United States using recordings from 2009 [27]. A nearest neighbors graph is constructed using longitudes and latitudes of the recording locations by connecting each station to its closest neighbors. Adding edges to each stations' eight closest neighbors was found empirically to construct a connected graph (Fig.3). The high-temperature readings from each location across the US on a single day are used to predict the next day's temperatures. We form our neural network from a series of quantum step layers (indicated by walk length) and a single diffusion layer. The QWNN uses the spatial coin setup and the unitary constraint on the coin operator is relaxed. For this experiment, we also compare the results with multiple DCNN walk lengths. The data is divided into thirds for training, validation, and testing and learning is limited to 128 epochs with early stopping if the validation score stops improving.

Table. 1 gives the test results for the trained networks. QW (A) and QW (C) learn only the initial walk amplitudes or the coin operators respectively while leaving the other fixed. The root mean squared prediction error and standard deviation are reported from five trials. We observe that quantum walk techniques yield lower errors compared to other graph neural network techniques. Learning both coin operators and initial amplitudes gives a small improvement over learning only one of the two and, perhaps surprisingly, learning only the initial amplitudes of the walk shows a marginal improvement over learning just the coin operators despite the coin operators having a larger potential effect on the walk. This may be due to the reduced number of parameters in the former making learning easier.

4.2 Graph Classification

The second type of graph problem we focus on is graph classification. We apply the graph neural networks to several common graph classification datasets: Enzymes [7], Mutag [11], and NCI1 [26]. Enzymes is a set of 600 molecules extracted from the Brenda database [23]. The task is to classify each enzyme into 1 of 6 classes. Mutag is a dataset

Table 1. Temperature Prediction Results

RMSE ± STD			
	Walk Length		
	2	3	4
GCN	8.56 ± 0.02		
DCNN	7.40 ± 0.13	7.46 ± 0.06	7.44 ± 0.10
GAT	7.84 ± 0.16		
QW (A+C)	5.54 ± 0.16	5.38 ± 0.07	$\mathbf{5.28 \pm 0.08}$
QW (A)	5.43 ± 0.15	5.43 ± 0.04	5.42 ± 0.23
QW (C)	6.05 ± 0.27	5.85 ± 0.07	5.97 ± 0.12

of 188 mutagenic aromatic and heteroaromatic nitro compounds that are classified into 1 of 2 categories, mutagenic or nonmutagenic. NCI1 consists of 4110 graphs representing two balanced subsets of datasets of chemical compounds screened for activity against nonsmall cell lung cancer. Summary statistics for each dataset are given in Table 2.

Each graph neural network consists of a graph layer, a dense layer with 10 hidden features and an output layer with a softmax activation. The QWNN uses a walk of length 4 and a single coin per timestep, while DCNN uses a 2-hop layer. Table 2 reports results for QWNN using the two different edge ordering given in Section 3.1. QWNNs demonstrate the highest mean accuracy on 2 out of the 3 datasets while underperforming the other methods on the Enzymes dataset.

Table 2. Graph Classification Datasets Summary and Results

	Enzymes	Mutag	NCI1
Graphs	600	188	4110
Average Nodes	33	18	30
Max Nodes	126	28	111
Max Degree	9	4	4
Node Classes	3	7	37
Graph Classes	6	2	2
	Classification Accuracy ± STD		
GCN	$\mathbf{0.41 \pm 0.08}$	0.83 ± 0.09	0.70 ± 0.02
DCNN	0.39 ± 0.05	0.86 ± 0.08	0.69 ± 0.01
GAT	0.34 ± 0.03	0.88 ± 0.035	0.70 ± 0.01
QWNN (centrality)	0.32 ± 0.03	0.85 ± 0.06	0.68 ± 0.02
QWNN (similarity)	0.32 ± 0.03	$\mathbf{0.92 \pm 0.03}$	$\mathbf{0.71 \pm 0.01}$

4.3 Graph Regression

Our graph regression task uses the QM7 dataset [6,22], a collection of 7165 molecules each containing up to 23 atoms. The geometries of these molecules are stored in Coulomb matrix format defined as

$$\mathbf{C}_{ij} = \begin{cases} 0.5Z_i^{2.4} & i = j \\ \frac{Z_i Z_j}{|R_i - R_j|} & i \neq j \end{cases}$$

where Z_i, R_i are the charge of and position of the i-th atom in the molecule respectively. The goal of the task is to predict the atomization energy of each molecule. Atomization energies of the molecules range from -440 to -2200 kcal/mol.

For this task, we form an approximation of the molecular graph from the Coulomb matrix by normalizing out the charges and separating the distances between atoms into 2 sets using K-means. One set contains the atom pairs with larger distances between them and the other the smaller distances. We create an adjacency matrix from all pairs of atoms in the smaller distance set. This creates connected graphs for all but 19 of the molecules. For the remaining 19 molecules, distant pairs of atoms are added iteratively until each molecular graph becomes fully connected. We take this approach to constructing the graphs because most molecules have a noticeable distance gap between directly bonded and unbonded pairs of atoms. We use the element of each atom, encoded as a one-hot vector, as the input features for each node.

The QWNN is composed of a four-step quantum random walk using a separate unitary coin at each step. Graphs smaller than the largest molecules (23 atoms) have both their feature and adjacency matrices appended with zeros to bring them up to size. For each graph neural network architecture, the graph layers feed two hidden layers of size 400 and 100 respectively and a final layer that outputs a single value with a sigmoid activation which is then rescaled to the range of atomization energies.

The RMSE and MAE (with standard deviations) are given in Table 3. QWNNs give noticeably better results over the other graph neural network approaches.

Table 3. Atomization Energy Prediction Results

	RMSE	*MAE*
GCN	19.26 ± 0.47	14.77 ± 0.26
DCNN	15.51 ± 1.12	10.73 ± 0.35
GAT	26.31 ± 4.09	19.64 ± 4.27
QWNN (centrality)	$\mathbf{12.52 \pm 0.91}$	$\mathbf{8.93 \pm 0.14}$
QWNN (similarity)	16.01 ± 0.96	10.78 ± 0.22

5 Limitations

Storing the superposition of a walker requires $O(Nd)$ space, with N the number of nodes in the graph, and d the max degree of the graph. To calculate a complete diffusion matrix requires a walker to begin at every node, and for the superposition of the walkers to be tracked for T time steps. This leads to a space requirement of $O(N^2dT)$ which is intractable for very large graphs, especially when doing learning on a GPU.

Quantum walks with shared coins across nodes are not invariant to the ordering of the edges to each node. In practice, we find that our heuristic ordering works well, Nonetheless, a QWNN can perform differently on isomorphic graphs. The spatial coin implementation of the QWNN does not inherit this problem because each coin can be tuned to the edge ordering specific to the node it operates at.

6 Concluding Remarks

Quantum walk neural networks offer a noticeable improvement over standard graph neural networks in both node and graph regression tasks and are competitive in graph classification tasks. QWNNs are able to adapt both spatially across the graph or temporally over the walk. The final diffusion matrix after learning has converged can be cached for quick forward passes in the network. However, the constant coin and shift operations during learning lead to a marked slowdown in our architecture compared to others, with space complexity also being an issue. In the current work, each walker on the graph operates independently. A future research direction is to investigate learning multi-walker quantum walks on graphs. Reducing the number of independent walkers and allowing interactions can reduce the space complexity of the quantum step layers.

References

1. Aharonov, D., Ambainis, A., Kempe, J., Vazirani, U.: Quantum walks on graphs. In: Proceedings of the thirty-third annual ACM symposium on Theory of computing, pp. 50–59. ACM (2001)
2. Ambainis, A.: Quantum walks and their algorithmic applications. International Journal of Quantum Information **1**(04), 507–518 (2003)
3. Ambainis, A., Bach, E., Nayak, A., Vishwanath, A., Watrous, J.: One-dimensional quantum walks. In: Proceedings of the thirty-third annual ACM symposium on Theory of computing, pp. 37–49. ACM (2001)
4. Atwood, J., Towsley, D.: Diffusion-convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1993–2001 (2016)
5. Bai, L., Rossi, L., Cui, L., Zhang, Z., Ren, P., Bai, X., Hancock, E.: Quantum kernels for unattributed graphs using discrete-time quantum walks. Pattern Recognition Letters **87**, 96–103 (2017)
6. Blum, L.C., Reymond, J.L.: 970 million druglike small molecules for virtual screening in the chemical universe database GDB-13. J. Am. Chem. Soc. **131**, 8732 (2009)
7. Borgwardt, K.M., Ong, C.S., Schönauer, S., Vishwanathan, S., Smola, A.J., Kriegel, H.P.: Protein function prediction via graph kernels. Bioinformatics **21**(suppl_1), i47–i56 (2005)
8. Brandes, U.: A faster algorithm for betweenness centrality. Journal of mathematical sociology **25**(2), 163–177 (2001)
9. Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and locally connected networks on graphs. In: International Conference on Learning Representations (ICLR2014), CBLS, April 2014 (2014)

10. Childs, A.M.: Universal computation by quantum walk. Physical review letters **102**(18), 180,501 (2009)
11. Debnath, A.K., Lopez de Compadre, R.L., Debnath, G., Shusterman, A.J., Hansch, C.: Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. Journal of medicinal chemistry **34**(2), 786–797 (1991)
12. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: Advances in Neural Information Processing Systems, pp. 3844–3852 (2016)
13. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems, pp. 1024–1034 (2017)
14. Jing, L., Shen, Y., Dubček, T., Peurifoy, J., Skirlo, S., Tegmark, M., Soljačić, M.: Tunable efficient unitary neural networks (eunn) and their application to rnn. arXiv preprint arXiv:1612.05231 (2016)
15. Kendon, V.: Quantum walks on general graphs. International Journal of Quantum Information **4**(05), 791–805 (2006)
16. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
17. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp. 1097–1105 (2012)
18. Monti, F., Bronstein, M., Bresson, X.: Geometric matrix completion with recurrent multi-graph neural networks. In: Advances in Neural Information Processing Systems, pp. 3697–3707 (2017)
19. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 701–710. ACM (2014)
20. Qiang, X., Yang, X., Wu, J., Zhu, X.: An enhanced classical approach to graph isomorphism using continuous-time quantum walk. Journal of Physics A: Mathematical and Theoretical **45**(4), 045,305 (2012)
21. Rohde, P.P., Schreiber, A., Štefaňák, M., Jex, I., Silberhorn, C.: Multi-walker discrete time quantum walks on arbitrary graphs, their properties and their photonic implementation. New Journal of Physics **13**(1), 013,001 (2011)
22. Rupp, M., Tkatchenko, A., Müller, K.R., von Lilienfeld, O.A.: Fast and accurate modeling of molecular atomization energies with machine learning. Physical Review Letters **108**, 058,301 (2012)
23. Schomburg, I., Chang, A., Ebeling, C., Gremse, M., Heldt, C., Huhn, G., Schomburg, D.: Brenda, the enzyme database: updates and major new developments. Nucleic acids research **32**(suppl_1), D431–D433 (2004)
24. Shenvi, N., Kempe, J., Whaley, K.B.: Quantum random-walk search algorithm. Physical Review A **67**(5), 052,307 (2003)
25. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
26. Wale, N., Watson, I.A., Karypis, G.: Comparison of descriptor spaces for chemical compound retrieval and classification. Knowledge and Information Systems **14**(3), 347–375 (2008)
27. Williams, C., Vose, R., Easterling, D., Menne, M.: United states historical climatology network daily temperature, precipitation, and snow data. ORNL/CDIAC-118, NDP-070. Available online [http://cdiac. ornl. gov/epubs/ndp/ushcn/usa. html] from the Carbon Dioxide Information Analysis Center, Oak Ridge National Laboratory, USA (2006)