

RESEARCH

Open Access



Quantum walk neural networks with feature dependent coins

Stefan Dernbach^{1*} , Arman Mohseni-Kabir¹, Siddharth Pal², Miles Gepner¹ and Don Towsley¹

*Correspondence:

dernbach@cs.umass.edu

¹University of Massachusetts,
College of Information and
Computer Sciences, Amherst 01003,
MA, USA

Full list of author information is
available at the end of the article

Abstract

Recent neural networks designed to operate on graph-structured data have proven effective in many domains. These graph neural networks often diffuse information using the spatial structure of the graph. We propose a quantum walk neural network that learns a diffusion operation that is not only dependent on the geometry of the graph but also on the features of the nodes and the learning task. A quantum walk neural network is based on learning the coin operators that determine the behavior of quantum random walks, the quantum parallel to classical random walks. We demonstrate the effectiveness of our method on multiple classification and regression tasks at both node and graph levels.

Keywords: Graph neural networks, Random walks, Quantum random walks

Introduction

While classical neural network approaches for structured data have been well investigated, there is growing interest in extending neural network architectures beyond grid structured data in the form of images or ordered sequences (Krizhevsky et al. 2012) to the domain of graph-structured data (Atwood and Towsley 2016; Bruna et al. 2014; Gori et al. 2005; Kipf and Welling 2016; Scarselli et al. 2009; Velickovic et al. 2017). Following the success of quantum kernels on graph-structured data Bai et al. (2013, 2017, 2015), a primary motivation of this work is to explore the application of quantum techniques and the potential advantages they might offer over classical algorithms. In this work, we propose a novel quantum walk based neural network structure that can be applied to graph data. Quantum random walks differ from classical random walks through additional operators (called coins) that can be tuned to affect the outcome of the walk.

In (Dernbach et al. 2018) we introduced a quantum walk neural network (QWNN) for the purpose of learning a task-specific random walk on a graph. When dealing with learning problems involving multiple graphs, the original QWNN formulation suffered from a requirement that all nodes across all graphs share the same coin matrix. This paper improves upon our original network architecture by replacing the single coin matrix with a bank that learns a function to produce different coin matrices at each node in every graph. This function allows the behavior of the quantum walk to vary spatially across the graph even when dealing with multi-graph problems. Additionally, this function produces the coins based on neighboring node features so that even for structurally identical graphs, a different walk is produced if the node features change. We also improve

the neural network architecture in this work. In the new architecture, each step of the quantum walk produces its own set of diffused features. The aggregated set of features, spanning the length of the walk, are passed to successive layers in the neural network. Finally, the previous work produced results that were dependent upon the ordering of the nodes. This work provides a QWNN architecture that is invariant to node ordering.

The rest of this paper is organized as follows. “[Related work](#)” section describes the background literature on graph neural network techniques in further detail. The setting of quantum walks on graphs is described in “[Graph quantum walks](#)” section, followed by a formal description of the proposed quantum walk based neural network implementation in “[Quantum walk neural networks](#)” section. Experimental results on node and graph regression, and graph classification tasks are presented in “[Experiments](#)” section, followed by a discussion of the techniques’ limitations in “[Limitations](#)” section and concluding remarks in “[Concluding remarks](#)” section.

Related work

Gupta and Zia (2001) and Altaisky (2001) among other researchers proposed quantum versions of artificial neural networks; See Biamonte et al. (2017) and Dunjko et al. (2018) for an overview of the emerging field of quantum machine learning. While not much work exists on quantum machine learning techniques for graph-structured data, in recent years, new neural network techniques that operate on graph-structured data have become prominent. Gori et al. (2005) followed by Scarselli et al. (2009) proposed recursive neural network architectures to deal with graph-structured data, instead of the then prevalent approach of transforming the graph data into a domain that could be handled by conventional machine learning algorithms. Bruna et al. (2014) studied the generalization of convolutional neural networks (CNNs) to graph signals through two approaches, one based upon hierarchical clustering of the domain, and another based on the spectrum of the graph Laplacian. Subsequently, Defferrard et al. (2016) proposed to approximate the convolutional filters on graphs through their fast localized versions.

Along with the spectral approaches described above, a number of spatial approaches have been proposed that relied on random walks to extract and learn information from the graph. For comparison, we detail several modern approaches. Atwood and Towsley (2016) propose a spatial convolutional method that performs random walks on the graph and combines information from spatially close neighbors. Given a graph $G = \{V, E\}$ and a feature matrix \mathbf{X} , their approach, Diffusion Convolutional Neural Networks (DCNN) use powers of the transition matrix $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$ to diffuse information across the graph, where \mathbf{A} is the adjacency matrix and \mathbf{D} is the diagonal degree matrix such that $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$. The k^{th} power of the transition matrix, \mathbf{P}^k , diffuses information from each node to every node exactly k hops away from it. The output \mathbf{Y} of the DCNN is a weighted combination of the diffused features from across the graph, given by

$$\mathbf{Y} = h(\mathbf{W} \odot \mathbf{P}^* \mathbf{X}),$$

where \mathbf{P}^* is the stacked tensor of powers of transition matrices, the operator \odot represents element-wise multiplication, \mathbf{W} are the learned weights of the diffusion-convolutional layer, and h is an activation function (e.g. rectified linear unit).

The second approach of interest due to Kipf and Welling (2016), was proposed to tackle semi-supervised learning on graph-structured data through a CNN architecture that

uses localized approximation of spectral graph convolutions. The proposed technique, the Graph Convolutional Neural Network (GCN) simplified the original spectral-based frameworks of Bruna et al. (2014) and Defferrard et al. (2016) for improved scalability. The method uses the augmented adjacency matrix $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and degree matrix $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$ to diffuse the input with respect to the local neighborhood according to:

$$\mathbf{Y} = h \left(\tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}} \mathbf{X} \mathbf{W} \right),$$

where, again, \mathbf{W} are learning weights and h is an activation function.

Many graph convolution layers are inspired by classical CNNs used in image recognition problems. However, other deep learning models have also inspired graph-based variants. One such example, Graph Attention Networks (GATs) (Velickovic et al. 2017), is inspired by the attention mechanisms commonly applied in natural language processing for sequence-based tasks. The neural network architecture uses a graph attention layer that combines information from neighboring nodes through an attention mechanism. Unlike the prior approaches, this allows a nonuniform weighting of the features of each node's neighbors. The method uses attention coefficients

$$e_{ij} = a(\mathbf{W}\mathbf{X}_i, \mathbf{W}\mathbf{X}_j)$$

where, \mathbf{W} is a learned weight matrix that linearly transforms feature vectors of nodes v_i and v_j , \mathbf{X}_i and \mathbf{X}_j respectively, and a is an attention function (e.g. inner product). The attention coefficients e_{ij} are normalized through the softmax function to obtain normalized coefficients α_{ij} . The output from node i is given as

$$\mathbf{Y}_i = h \left(\sum_{v_j \in \mathcal{N}(v_i)} \alpha_{ij} \mathbf{W}\mathbf{X}_j \right)$$

where $\mathcal{N}(v_i)$ is the neighbor set of node v_i .

Our proposed quantum walk neural network is a graph neural network architecture based on discrete quantum walks. Various researchers have worked on quantum walks on graphs – Ambainis et al. (2001) studied quantum variants of random walks on one-dimensional lattices; Farhi and Gutmann (1998) reformulated interesting computational problems in terms of decision trees, and devised quantum walk algorithms that could solve problem instances in polynomial time compared to classical random walk algorithms that require exponential time. Aharonov et al. (2001) generalized quantum walks to arbitrary graphs. Subsequently, Rohde et al. (2011) studied the generalization of discrete time quantum walks to the case of an arbitrary number of walkers acting on arbitrary graph structures, and their physical implementation in the context of linear optics. Quantum walks have recently become the focus of many graph-analytics studies because of their non-classical interference properties. Bai et al. (2013, 2017, 2015) introduced novel graph kernels based on the evolution of quantum walks on graphs. They defined the similarity between two graphs in terms of the similarities between the evolution of quantum walks on the two graphs. Quantum kernel based techniques were shown to outperform classical kernel techniques in effectiveness and accuracy. In Rossi et al. (2013, 2015), Rossi et al. studied the evolution of quantum walks on the union of two graphs to define the kernel between two graphs. These closely related works on quantum walks and the success of quantum kernel techniques motivated our approach in developing a quantum neural network architecture.

Graph quantum walks

Motivated by classical random walks, quantum walks were introduced by (Aharonov et al. 1993). Unlike the stochastic evolution of a classical random walk, a quantum walk evolves according to unitary process. The behavior of a quantum walk is fundamentally different from a classical walk since in a quantum walk there is interference between different trajectories of the walk. Two kinds of quantum walks have been introduced in the literature; namely, continuous time quantum walks (Farhi and Gutmann 1998; Rossi et al. 2017) and discrete time quantum walks (Lovett et al. 2010). Quantum walks have recently received much attention because they have been shown to be a universal model for quantum computation (Childs 2009). In addition, they have numerous applications in quantum information science such as database search (Shenvi et al. 2003), graph isomorphism (Qiang et al. 2012), network analysis and navigation, and quantum simulation.

Discrete time quantum walks were initially introduced on simple regular lattices (Nayak and Vishwanath 2000) and then extended to general graphs (Kendon 2006). In this paper, we use the formulation of discrete time quantum walks as outlined in (Ambainis 2003; Kendon 2006). Given an undirected graph $G = (V, E)$, we introduce a position Hilbert space \mathcal{H}_P that captures the superposition over various positions, i.e., nodes, in the graph. We define \mathcal{H}_P to be the span of the position basis vectors $\{\hat{\mathbf{e}}_v^{(p)}, v \in V\}$. The position vector of a quantum walker can now be written as a linear combination of position state basis vectors,

$$\psi_p = \sum_{v \in V} \alpha_v \hat{\mathbf{e}}_v^{(p)}$$

where $\{\alpha_v, v \in V\}$ are coefficients satisfying the unit L_2 -norm condition $\sum_v \|\alpha_v\|^2 = 1$, with the understanding that $\|\alpha_v\|^2$ is the probability of finding the walker at vertex v .

Similarly, we introduce a coin Hilbert space \mathcal{H}_C that captures the superposition over various spin directions of the walker on each node of the graph. We define \mathcal{H}_C to be the span of the coin basis vectors $\{\hat{\mathbf{e}}_i^{(c)}, i \in 1, \dots, d_{max}\}$, where i enumerates the edges incident on a vertex v and d_{max} is the maximum degree of the graph. We will use d instead of d_{max} for conciseness. The coin (spin) state of a quantum walker can now be written as a linear combination of coin state basis vectors,

$$\psi_c = \sum_{i \in 1, \dots, d} \beta_{v,i} \hat{\mathbf{e}}_i^{(c)}$$

where $\{\beta_{v,i}, i \in 1, \dots, d\}$ are coefficients satisfying the unit L_2 -norm condition $\sum_i |\beta_{v,i}|^2 = 1$. If a measurement is done on the coin state of the walker at vertex v , $|\beta_{v,i}|^2$ denotes the probability of finding the walker in coin state i . The Hilbert space of the quantum walk can be written as $\mathcal{H}_W = \mathcal{H}_P \otimes \mathcal{H}_C$, which is the tensor product of the two aforementioned Hilbert spaces.

Time-evolution of discrete time quantum walk over graph G is governed by two unitary operators, namely, coin and shift operator. Let $\Phi^{(t)} = \psi_p^{(t)} \otimes \psi_c^{(t)}$ in \mathcal{H}_W denote the state of the walker at time t . At each time-step we first apply a unitary coin operator \mathbf{C} which transforms the coin state of the walker at each vertex,

$$\psi_p^{(t)} \otimes \psi_c^{(t+1)} = (\mathbf{I} \otimes \mathbf{C}) \left(\psi_p^{(t)} \otimes \psi_c^{(t)} \right).$$

\mathbf{I} denotes the identity operator. After transforming the coin (spin) states, we apply a unitary shift operator \mathbf{S} which 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 we swap the coefficient corresponding to the basis state $\hat{e}_v^{(p)} \otimes \hat{e}_i^{(c)}$ with that of the basis state $\hat{e}_u^{(p)} \otimes \hat{e}_j^{(c)}$. S operates on both coin and position Hilbert spaces,

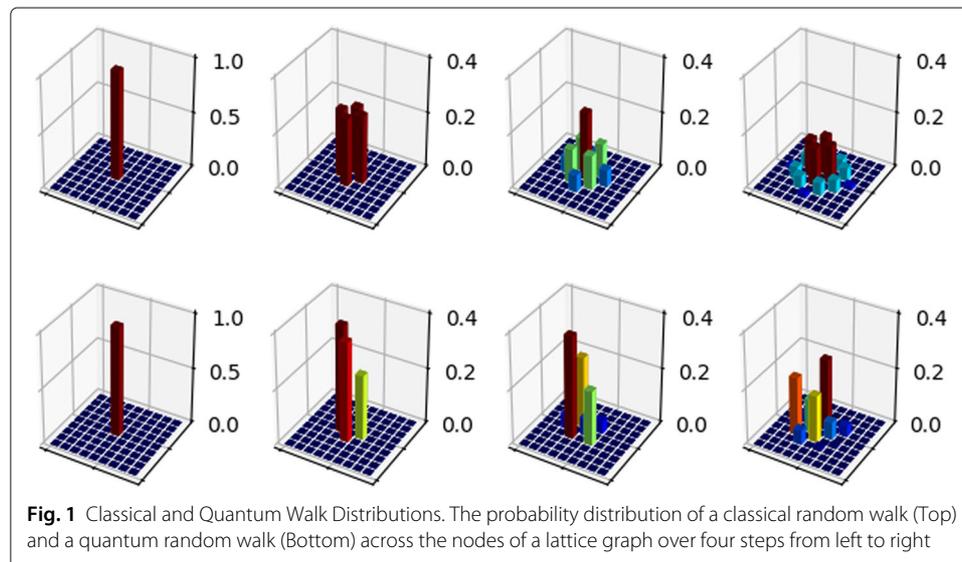
$$\Phi^{(t+1)} = \psi_p^{(t+1)} \otimes \psi_c^{(t+1)} = S(\psi_p^{(t)} \otimes \psi_c^{(t+1)}).$$

In shorthand notation, the unitary evolution of the walk is governed by the operator $U = S(I \otimes C)$. Applying U successively evolves the state of the quantum walk through time.

The choice of coin operators as well as the initial superposition of the walker control how this non-classical diffusion process evolves over the graph and therefore provides the deep learning technique additional degrees of freedom for controlling the flow of information over the graph. Figure 1 shows how the diffusion behavior of a classical random walk differs from a discrete time quantum walk with a single coin. Ahmad et al. (2019) recently showed that for a discrete quantum walk on a line, having a position-dependent coin can lead to quantitatively different diffusion behaviors with different choices of coin operators. Our work uses the setting of multiple non-interacting quantum walks acting on arbitrary graphs, as introduced in Rhode et al. (2011), to learn patterns in graph data. Calculating a separate quantum walk originating from each node in the graph allows us to construct a diffusion matrix where each entry gives the relationship between the starting and ending nodes of a walk. This matrix works like its classical counterpart, a random walk matrix, used in DCNN (Atwood and Towsley 2016).

Physical implementation of discrete quantum walks

Over the past few years, there have been several proposals for the physical implementation of quantum walks. Quantum walks are unitary process that are naturally implementable in a quantum system by manipulating their internal structure. The internal structure of the quantum system should be engineered to be able to manifest the position and coin Hilbert spaces of the quantum walk. These quantum simulation based methods have been proposed using classical and quantum optics (Zhang et al. 2007), nuclear magnetic resonance (Ryan et al. 2005), ion traps (Travaglione and Milburn 2002), cavity QED



(Agarwal and Pathak 2005), optical lattices (Joo et al. 2007), and Bose Einstein condensate (Manouchehri and Wang 2009) as well as quantum dots (Manouchehri and Wang 2008) to implement the quantum walk.

Circuit implementation of quantum walks has also been proposed. While most of these implementations focus on graphs that have a very high degree of symmetry (Loke and Wang 2011) or very sparse graphs (Jordan and Wocjan 2009; Chiang et al. 2010), there is some recent work on circuit implementations on non-degree regular graphs (Loke and Wang 2012).

A central question in implementing quantum walks on graphs is how to scale the physical system to achieve the complexity required for simulating large graphs. Rohde et al. (2013) showed that exponentially larger graphs can be constructed using quantum entanglement as a resource for creating very large Hilbert spaces. They use multiple entangled walkers to simulate a quantum walk on a virtual graph of chosen dimensions. However, this approach has its own limitations and arbitrary graphs can not be built with this method.

Quantum walk neural networks

Many graph neural networks pass information between two nodes based on the distance between the nodes in the graph. This is true for both graph convolution networks and diffusion convolution networks. However, quantum walk neural networks are similar to graph attention networks in that the amount of information passed between two nodes also depends on the features of the nodes. In graph attention networks this is achieved by calculating an attention coefficient for each of a node’s neighbors. In quantum walk neural networks, the coin operator alters the spin states of the quantum walk to prioritize specific neighbors.

A QWNN, as shown in Fig. 2, learns a quantum walk on a graph by means of back propagating gradient updates to the coin operators used in the walk. The learned walk is then used to diffuse a signal over the graph.

In (Dernbach et al. 2018), the quantum walk neural network evolves a walk using a single coin matrix, C , to modify the spin state of the walker Φ according to $\Phi^{(t+1)} = \Phi^{(t)}C^{(t)}$

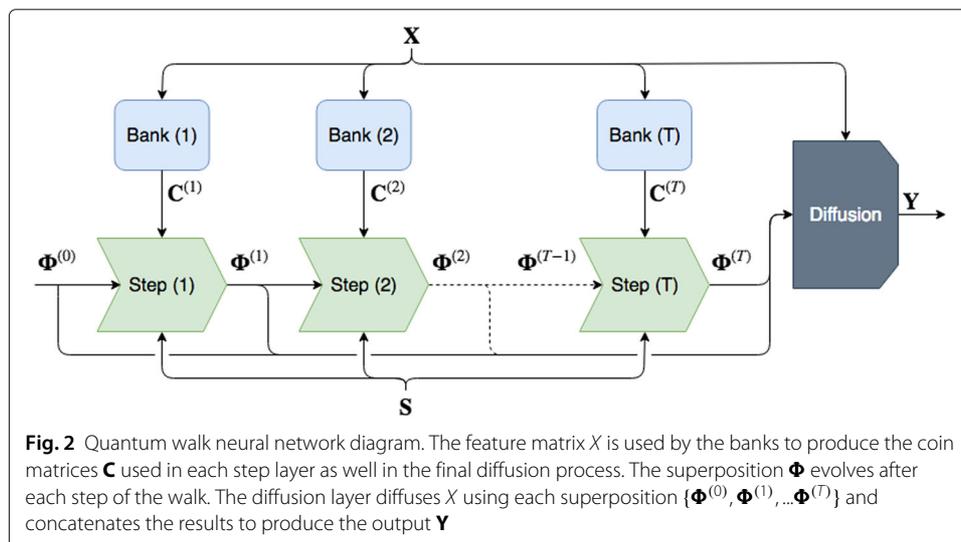


Fig. 2 Quantum walk neural network diagram. The feature matrix X is used by the banks to produce the coin matrices C used in each step layer as well in the final diffusion process. The superposition Φ evolves after each step of the walk. The diffusion layer diffuses X using each superposition $\{\Phi^{(0)}, \Phi^{(1)}, \dots, \Phi^{(T)}\}$ and concatenates the results to produce the output Y

and then swaps states along the edges of the graph. Features are then diffused across the graph by converting the states of the walker into a probability matrix, \mathbf{P} , and using it to diffuse the feature matrix: $\mathbf{Y} = \mathbf{P}\mathbf{X}$. The coin matrix is learned through backpropagating the gradient of a loss function. In this paper we replace the coin matrix by a node and time dependent function we call a bank. The bank forms the first of the three primary parts of a QWNN. It is followed by the walk and the diffusion. The bank produces the coin matrices used to direct the quantum walk, the walk layers determine the evolution of the quantum walk at each step, and the diffusion layer uses these states to spread information throughout the graph.

Bank

The Coin operators modify the spin state of the walk and are thus the primary levers by which a quantum walk is controlled. The coin operator can vary spatially across nodes in the graph, temporally along steps of the walk, or remain constant in either or both dimensions. In the QWNN, the bank produces these coins for the quantum walk layers.

When the learning environment is restricted to a single static graph, the bank stores the coin operators as individual coin matrices distributed across each node in the graph. However, for dynamic or multi-graph situations, the bank operates by learning a function that produces coin operators from node features $f : X \rightarrow \mathbb{C}^{d \times d}$ where d is the maximum degree of the graph. In general, f is any arbitrary function that produces a matrix followed by a unitary projection to produce a coin \mathbf{C} . This projection step is expensive as it requires a singular value decomposition of a $d \times d$ matrix.

In recurrent neural networks (RNN), unitary matrices are employed to deal with exploding or vanishing gradients because backpropagating through a unitary matrix does not change the norm of the gradient. To avoid expensive unitary projections, several recursive neural network architectures use functions f whose ranges are subsets of unitary matrices. A common practice is to use combinations of low dimensional rotation matrices (Arjovsky et al. 2016; Jing et al. 2017). This was the model used for the coin operators in previous QWNNs (Dernbach et al. 2018).

In our work, we focus on elementary unitary matrices. These matrices are of the form $\mathbf{U} = \mathbf{I} - 2\mathbf{w}\mathbf{w}^T / (\mathbf{w}^T \mathbf{w})$ where \mathbf{I} denotes the identity matrix and \mathbf{w} is any vector. These matrices can be computed efficiently in the forward pass of the neural network and their gradients can similarly be computed efficiently during backpropagation. While this work focuses on using a single elementary matrix for each coin operator, any unitary matrix can be composed as the product of elementary unitary matrices. The QWNN bank produces the coin matrix for node v_i according the following:

$$\mathbf{C}_i = \mathbf{I} - 2f(v_i)f(v_i)^T / (f(v_i)^T f(v_i)).$$

We propose two different functions $f(v_i)$.

The first function:

$$f_1(v_i) = \mathbf{W}^T \text{vec}(\mathbf{X}_{\mathcal{N}(v_i)}) + \mathbf{b},$$

where $\text{vec}(\mathbf{X}_{\mathcal{N}(v_i)})$ denotes the column vector of concatenated features of the neighbors of v_i , is a standard linear function parameterized by a weight matrix $\mathbf{W} \in \mathbb{R}^{(Fd) \times d}$, with F the number of features, and a bias vector $\mathbf{b} \in \mathbb{R}^d$. This method has individual weights for each node but is not equivariant to the ordering of the nodes in the graph. This means

that permuting the neighbors of v_i changes the result of the function. We mitigate this effect by using a heuristic node ordering based on node centrality that we outline in “Node and neighborhood ordering” section.

The second function:

$$f_2(v_i) = \mathbf{X}_{\mathcal{N}(i)} \mathbf{W} \mathbf{X}_i^T,$$

with $\mathbf{W} \in \mathbb{R}^{F \times F}$, computes a similarity measure between the node v_i and each of its neighbors. This method is equivariant with respect to the node ordering of the graph (i.e. permuting the neighborhood of v_i equally permutes the values of $f_k(v_i)$). This in turn allows the entire neural network to be invariant to node ordering.

Walk

For a graph with N vertices, the QWNN processes N separate, non-interacting walks in parallel – one walk originating from each node in the graph. The walks share the same bank functions. A T -step walk produces a sequence of superpositions $\{\Phi^{(0)}, \Phi^{(1)}, \dots, \Phi^{(T)}\}$. For a graph with degree d , the initial superposition tensor $\Phi^{(0)} \in \mathbb{C}^{N \times N \times d}$ is initialized with equal spin along all incident edges to the node it begins at such that $(\Phi_{ii}^{(0)})^H \Phi_{ii}^{(0)} = 1$ and $\forall i \neq j : \Phi_{ijk}^{(0)} = 0$. The value of $\Phi_{ijk}^{(t)}$ denotes the amplitude of the i -th walker at node v_j with spin k after t steps of the walk.

A complete walk can be broken down into individual step layers. Each quantum step layer takes as input the current superposition tensor $\Phi^{(t)}$, the set of coins operators $\mathbf{C}^{(t)}$ produced by the bank, as well as a shift tensor $\mathbf{S} \in \mathbb{Z}_2^{N \times d \times N \times d}$ that encodes the graph structure: $S_{ijvi} = 1$ iff u is the i^{th} neighbor of v and v is the j^{th} neighbor of u . The superposition evolves according to:

$$\Phi^{(t+1)} = \Phi^{(t)} \mathbf{C}^{(t)} \mathbf{S}$$

where $\mathbf{A} \cdot \mathbf{B}$ denotes the tensor double inner product of \mathbf{A} and \mathbf{B} . Equivalently, for an edge (u, v) , with u being the i^{th} neighbor of v and v being the j^{th} neighbor of u :

$$\begin{aligned} \Phi_{wuj}^{(t+1)} &= \left(\Phi_v^{(t)} \mathbf{C}_v^{(t)} \right)_{wi} \\ \Phi_{wvi}^{(t+1)} &= \left(\Phi_u^{(t)} \mathbf{C}_u^{(t)} \right)_{wj} \end{aligned}$$

The output $\Phi^{(t+1)}$ is fed into the next quantum step layer (if there is one) and the diffusion layer.

Diffusion

The superpositions at each step of the walk are used to diffuse the signal \mathbf{X} across the graph. Given a superposition Φ , the diffusion matrix is constructed by summing the squares of the spin states: $\mathbf{P} = \sum_k \Phi_{..k} \odot \Phi_{..k}$. The value \mathbf{P}_{ij} gives the probability of the walker beginning at v_i and ending at v_j similar to a classical random walk matrix. Diffused features can then be computed as a function of \mathbf{P} and \mathbf{X} by $\mathbf{Y} = h(\mathbf{P}\mathbf{X} + \mathbf{b})$ where h is an optional nonlinearity (e.g. ReLU). The complete calculation for a forward pass for the QWNN is given in Algorithm 1.

Algorithm 1: QWNN Forward Pass	
given	: Initial Superpositions $\Phi^{(0)}$, Shift S
input	: Features X
output	: Diffused Features Y
1	for $t = 1$ to T do
2	for All nodes v_i do
3	$\mathbf{v}_i^{(t)} = \mathbf{W}^T \text{vec}(X_{N(v_i)}) + \mathbf{b}$ or $\mathbf{v}_i^{(t)} = X_{N(i)} \mathbf{W} X_i^T$
4	$\mathbf{C}_i^{(t)} = \mathbf{I} - 2\mathbf{v}_i^{(t)} (\mathbf{v}_i^{(t)})^T / ((\mathbf{v}_i^{(t)})^T \mathbf{v}_i^{(t)})$
5	$\Phi_{\cdot i}^{(t)} \leftarrow \Phi_{\cdot i}^{(t-1)} \cdot \mathbf{C}_i^{(t)}$
6	$\Phi^{(t)} \leftarrow \Phi^{(t)} \cdot S$ (i.e., $\Phi_{wuj}^{(t)} = \sum_v \sum_i \Phi_{wvi}^{(t)} S_{viju}$)
7	$\mathbf{P}^{(t)} \leftarrow \sum_k \Phi_{\cdot k}^{(t)} \odot \Phi_{\cdot k}^{(t)}$
8	$\mathbf{Y}^{(t)} \leftarrow h(\mathbf{P}^{(t)} X + \mathbf{b}^{(t)})$
	return : $\{\mathbf{Y}^{(0)}, \mathbf{Y}^{(1)}, \dots, \mathbf{Y}^{(T)}\}$

Node and neighborhood ordering

Node ordering and by extension neighborhood ordering of each node can have an effect on a quantum walk if the coin is not equivariant to the ordering. Given a non-equivariant set of coins, if the order of nodes in the graph is permuted, the result of the walk may change.

This is the case for the first of the two bank functions. We address this issue using a centrality score. The betweenness centrality (Brandes 2001) 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 . A larger betweenness centrality score implies a node is more central within the graph. Conversely, a leaf node connected to the rest of the graph by a single edge has a score of 0. Nodes in the graph are then ranked by their betweenness centrality and each neighborhood follows this ranking so that when ordering a node's neighbors, the most central nodes in the graph come first. 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.

Experiments

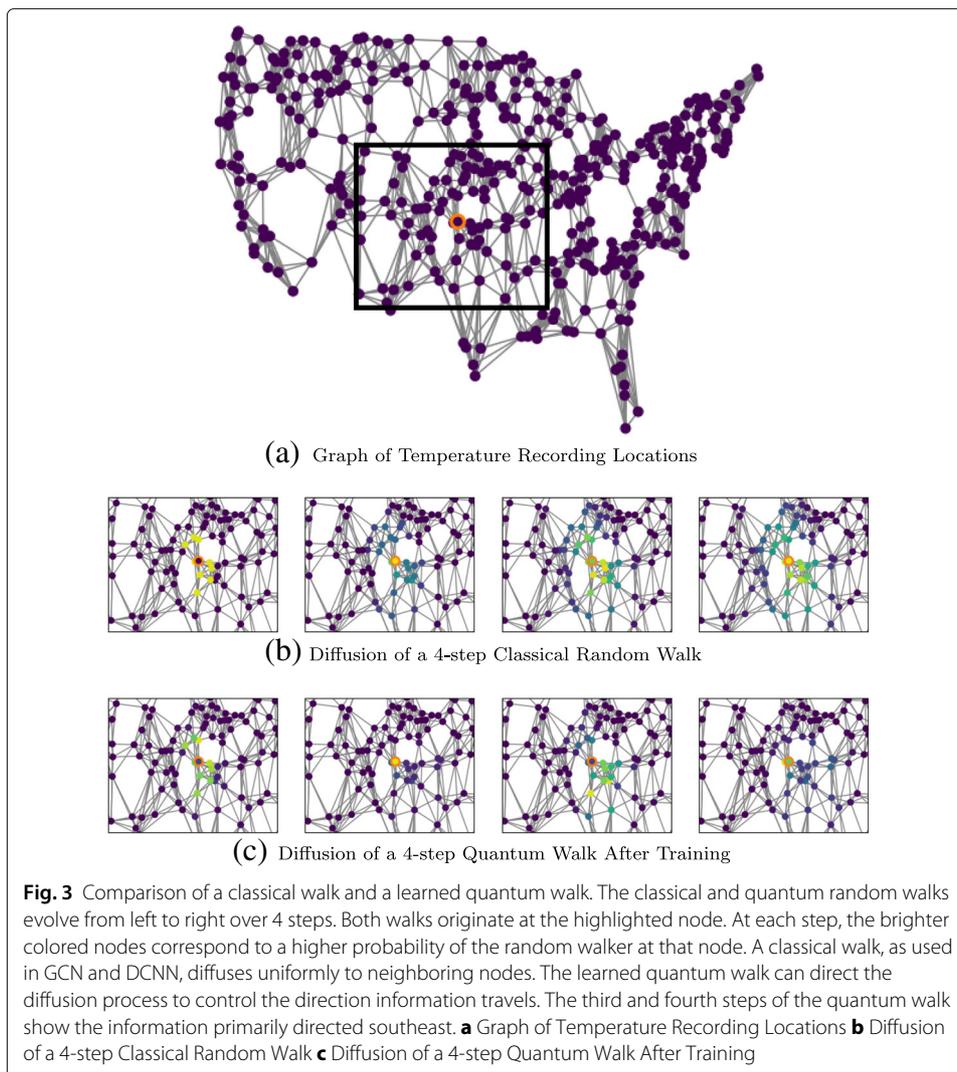
We demonstrate the effectiveness of QWNNs across three different types of tasks: node level regression, graph classification and graph regression. Our experiments focus on comparisons with three other graph neural network architectures: diffusion convolution neural networks (DCNN) (Atwood and Towsley 2016), graph convolution networks (GCN) (Kipf and Welling 2016), and graph attention networks (GAT) (Velickovic et al. 2017).

For graph level experiments, we employ a set2vec layer (Vinyals et al. 2016) as an intermediary between the graph layers and standard neural network feed forward layers. Set2vec has proved effective in other graph neural networks (Gilmer et al. 2017) as it is

a permutation invariant function that converts a set of node features into a fixed length vector.

Node regression

In the node regression task, daily temperatures are recorded across 409 locations in the United States during the year 2009 (Williams et al. 2006). The goal of the task is to use a day's temperature reading to predict the next day's temperatures. A nearest neighbors graph (Fig. 3a) is constructed using longitudes and latitudes of the recording locations by connecting each station to its closest neighbors. Adding edges to each station's eight closest neighbors produces a connected graph. The QWNN is formed from a series of quantum step layers (indicated by walk length) followed by a diffusion layer. Since the neural network in this experiment only uses quantum walk layers, we relax the unitary constraint on the coin operators. While this can no longer be considered a quantum walk in the strictest sense, the relaxation is necessary to allow the temperature vector to grow or shrink to match increases or decreases in temperatures from day to day. For this experiment, we also compare the results with multiple DCNN walk lengths. For GCN and GAT



an effective walk length is constructed by stacking layers. Data is divided into thirds for training, validation, and testing. Learning is limited to 32 epochs.

Table 1 gives the test results for the trained networks. The root-mean-square error (RMSE) and standard deviation (STD) are reported from five trials. We observe that quantum walk techniques yield lower errors compared to other graph neural network techniques. The two networks which control the amount of information flow between nodes, QWNN and GAT, appear to be able to take advantage of more distant relationships in the graph for learning while DCNN and GCN perform best with more restrictive neighborhood sizes.

We use this experiment to provide a visualization for the learned quantum walk. Figure 3b and c shows the evolution of a classical random walk and the learned quantum random walk originating from the highlighted node respectively. At each step, warmer color nodes correspond to nodes with higher superposition amplitudes. Initially, the quantum walk appears to diffuse outward in a symmetrical manner similar to a classical random walk, but in the third and fourth steps of the walk, the learned quantum walk focuses information flow towards the southeast direction. The ability to direct the walk in this way proves beneficial in the prediction task.

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 (Borgwardt et al. 2005), Mutag (Debnath et al. 1991), and NCI1 (Wale et al. 2008). Enzymes is a set of 600 molecules extracted from the Brenda database (Schomburg et al. 2004). In the dataset, each graph represents a protein and each node represents a secondary structure element (SSE) within the protein structure, e.g. helices, sheets and turns. Nodes are connected if certain conditions are satisfied, with each node bearing a type label, and its physical and chemical information. The task is to classify each enzyme into one of six classes. Mutag is a dataset of 188 mutagenic aromatic and heteroaromatic nitro compounds that are classified into one of two categories based on whether they exhibit a mutagenic effect. NCI1 consists of 4110 graphs representing two balanced subsets of chemical compounds screened for activity against non-small cell lung cancer. For both the Mutag and NCI1 datasets, each graph represents a molecule, with nodes representing atoms and edges representing bonds between atoms. Each node has an associated label that corresponds to its atomic number. Summary statistics for each dataset are given in Table 2. The experiments are run using 10-fold cross validation.

For the Enzyme and NCI1 experiment, the quantum walk neural networks are composed of a length 6 walk, followed by a set2vec layer, a hidden layer of size 64,

Table 1 Temperature prediction results

RMSE \pm STD					
Walk Length	1	2	3	4	5
GCN	8.56 \pm 0.02	8.14 \pm 0.41	7.82 \pm 0.13	8.55 \pm 0.52	8.88 \pm 0.73
DCNN	8.07 \pm 0.21	7.40 \pm 0.13	7.46 \pm 0.06	7.44 \pm 0.10	10.19 \pm 0.18
GAT	7.84 \pm 0.16	8.43 \pm 0.42	8.47 \pm 1.02	8.23 \pm 0.69	7.93 \pm 0.15
QWNN	6.11 \pm 0.14	5.54 \pm 0.16	5.38 \pm 0.07	5.28 \pm 0.08	5.65 \pm 0.02

The best performing network is in bold

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 \pm STD		
GCN	0.31 \pm 0.06	0.87 \pm 0.10	0.69 \pm 0.02
DCNN	0.27 \pm 0.08	0.89 \pm 0.10	0.69 \pm 0.01
GAT	0.32 \pm 0.04	0.89 \pm 0.06	0.66 \pm 0.03
QWNN (cen)	0.26 \pm 0.03	0.90 \pm 0.09	0.76 \pm 0.01
QWNN (inv)	0.33 \pm 0.04	0.88 \pm 0.04	0.73 \pm 0.02
WL	0.59 \pm 0.01	0.84 \pm 0.01	0.85 \pm 0.00
SP	0.41 \pm 0.02	0.87 \pm 0.01	0.73 \pm 0.00

The best performing neural networks and the overall best performing methods are bolded for each task

and a final softmax layer. In Mutag, the walk length is reduced to 4 and the hidden layer to 16. The reduced size helps alleviate some of the overfitting from such a small training set. We report the best results using the centrality based node ordering version of the network that uses the linear bank function: QWNN (cen) as well as the invariant QWNN using the equivariant bank function: QWNN (inv). We also report results from the three other graph networks. GCN, DCNN, and GAT are all used as an initial layer to a similar neural network followed by a set2vec layer, a hidden layer of size 64 (16 for Mutag) and a softmax output layer. DCNN uses a walk length of 2, while GCN and GAT use feature sizes of 32. Additionally we compare with two graph kernel methods, Weisfeiler-Lehman (WL) kernels (Shervashidze et al. 2011) and shortest path (SP) kernels (Borgwardt and Kriegel 2005), using the results given in (Shervashidze et al. 2011).

Classification accuracies are reported in Table 2. The best neural network accuracies and the best overall accuracies are bolded. Quantum Walks are competitive with the other neural network approaches. QWNN demonstrates the best average accuracy on Mutag and Enzyme but the other neural network approaches are within the margin of error. On the NCI1 experiment, QWNN shows a measurable improvement over the other neural networks. The WL kernels outperform all the neural network approaches on both Enzymes and NCI1.

Graph regression

Our graph regression task uses the QM7 dataset (Blum and Raymond 2009; Rupp et al. 2012), a collection of 7165 molecules each containing up to 23 atoms. The geometries of these molecules are stored in Coulomb matrix format defined as

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

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 atomic charges and separating all atom-atom pairs into two sets based on their physical distances. 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. There is generally a significant gap between the distances of bonded and unbonded atoms in a molecule but this approach leaves 19 disconnected graphs. For these molecules, edges are added between the least distant pairs of atoms until the graph becomes connected. We use the element of each atom, encoded as a one-hot vector, as the input features for each node.

The two variants of QWNN are constructed using a 4-step walk, followed by the set2vec layer, a hidden layer of size 10, and a final output layer. For the other graph neural networks, a single graph layer is used followed by the same setup of a set2vec layer, a hidden layer of size 10, and the output layer. A DCNN of length 2 walk and GCN and GAT using 32 features were found to give the best results. Root-mean-square error (RMSE) and mean absolute prediction error (MAE) are reported for each network in Table 3. QWNNs demonstrate a marked improvement over other methods in this task.

Limitations

Storing the superposition of a single 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 that a separate walker begin at every node, increasing the space requirement to $O(N^2d)$ which starts to become intractable for very large graphs, especially when doing learning on a graphics processing unit (GPU). Some of this cost can be alleviated using sparse tensors. At time $t=0$ the superpositions are localized to single nodes so only $O(Nd)$ space used by nonzero amplitudes. At time $t=1$ the first step increases this to $O(Nd^2)$ as each neighboring node becomes nonzero. Given a function $s(G, t)$ which determines the number of nodes in a graph reachable after a t -length random walk, the space complexity for a t -length walk is $O(Nds(G, t))$.

The majority of graph neural networks are invariant to the ordering of the nodes in the graph. This is true for GCN, DCNN, and GAT. We provide one formulation for a QWNN that is also invariant, however the second formulation is not. Although we have greatly reduced the effect, node ordering can still affect the walk produced in QWNN and thus the overall output of the network. This can occur when two otherwise distinguishable nodes have the same betweenness centrality.

Concluding remarks

Quantum walk neural networks provide a unique neural network approach to graph classification and regression problems. Unlike prior graph neural networks, QWNNs fully

Table 3 Atomization energy prediction results

	RMSE	MAE
GCN	16.51 ± 0.38	12.39 ± 0.29
DCNN	11.90 ± 0.59	8.53 ± 0.42
GAT	18.75 ± 0.51	14.52 ± 1.12
QWNN (cen)	9.70 ± 0.77	6.74 ± 0.24
QWNN (inv)	10.91 ± 0.56	8.28 ± 0.47

The best performing network is in bold

integrate the graph structure and the graph signal into the learning process. This allows QWNN to learn task dependent walks on complex graphs. The benefit of using the distributions produced by these walks as diffusion operators is especially clear in regression problems where QWNN demonstrate considerable improvement over other graph neural network approaches. This improvement is demonstrated at both the node and the graph level.

An added benefit of QWNN is that the learned walks provide a human understandable glimpse of the neural network determination of where information originating from each node is most beneficial in the graph. 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 walk layers.

Abbreviations

CNN: Convolutional neural networks; DCNN: Diffusion convolutional neural network; GAT: Graph attention network; GCN: Graph convolutional neural network; GPU: Graphics processing unit; MAE: Mean absolute error; QWNN: Quantum walk neural networks; RMSE: Root mean squared prediction error; RNN: Recursive neural network; SP: Shortest path; STD: Standard deviation; WL: Weisfeiler-Lehman

Acknowledgements

Not applicable.

Authors' contributions

SD worked on conceptualization, methodology, software writing, experiments, writing, and review and editing of the paper. AMK worked on conceptualization, writing, and review and editing of the paper. SP worked on conceptualization, writing, review and editing of the paper, and acquisition of funding for the research. MG helped with the methodology and worked on software. DT worked on conceptualization, review and editing, supervision of the research, acquisition of funding, and methodology. All authors read and approved the final manuscript.

Funding

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.

Availability of data and materials

The US Temperature dataset (Williams et al. 2006) was compiled from recordings prepared by the Carbon Dioxide Information Analysis Center and is available at <http://cdiac.ornl.gov/epubs/ndp/ushcn/usa.html>. The Mutag (Debnath et al. 1991), Enzymes (Borgwardt et al. 2005), and NCI1 (Wale et al. 2008) datasets are part of the benchmark datasets for graph kernels available at <https://ls11-www.cs.tu-dortmund.de/staff/morris/graphkerneldatasets>. The QM7 dataset (Blum and Raymond 2009; Rupp et al. 2012) is available at <http://quantum-machine.org/datasets/>.

Competing interests

The authors declare that they have no competing interests.

Author details

¹University of Massachusetts, College of Information and Computer Sciences, Amherst 01003, MA, USA. ²Raytheon BBN Technologies, Cambridge 02138, MA, USA.

Received: 17 March 2019 Accepted: 12 August 2019

Published online: 23 September 2019

References

- Agarwal GS, Pathak PK (2005) Quantum random walk of the field in an externally driven cavity. *Phys Rev A* 72(3):033815
- Aharonov Y, Davidovich L, Zagury N (1993) Quantum random walks. *Phys Rev A* 48(2):1687
- Aharonov D, Ambainis A, Kempe J, Vazirani U (2001) Quantum Walks on Graphs. In: Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing. ACM, New York, pp 50–59
- Ahmad R, Sajjad U, Sajid M (2019) One-dimensional quantum walks with a position-dependent coin. arXiv preprint arXiv:1902.10988
- Altshuler M (2001) Quantum neural network. arXiv preprint quant-ph/0107012
- Ambainis A (2003) Quantum walks and their algorithmic applications. *Int J Quantum Inf* 1(04):507–518

- Ambainis A, Bach E, Nayak A, Vishwanath A, Watrous J (2001) One-dimensional Quantum Walks. In: Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing. ACM, New York. pp 37–49
- Arjovsky M, Shah A, Bengio Y (2016) Unitary evolution recurrent neural networks. In: International Conference on Machine Learning. pp 1120–1128
- Atwood J, Towsley D (2016) Diffusion-Convolutional Neural Networks. In: Advances in Neural Information Processing Systems 29. Curran Associates, Inc., Red Hook. pp 1993–2001
- Bai L, Hancock ER, Torsello A, Rossi L (2013) A quantum jensen-shannon graph kernel using the continuous-time quantum walk. In: International Workshop on Graph-Based Representations in Pattern Recognition. Springer, Berlin. pp 121–131
- Bai L, Rossi L, Cui L, Zhang Z, Ren P, Bai X, Hancock E (2017) Quantum kernels for unattributed graphs using discrete-time quantum walks. *Pattern Recogn Lett* 87:96–103
- Bai L, Rossi L, Torsello A, Hancock ER (2015) A quantum jensen–shannon graph kernel for unattributed graphs. *Pattern Recogn* 48(2):344–355
- Biamonte J, Wittek P, Pancotti N, Rebentrost P, Wiebe N, Lloyd S (2017) Quantum machine learning. *Nature* 549(7671):195
- Blum LC, Reymond J-L (2009) 970 million druglike small molecules for virtual screening in the chemical universe database GDB-13. *J Am Chem Soc* 131:8732
- Borgwardt KM, Kriegel H-P (2005) Shortest-path kernels on graphs. In: Fifth IEEE International Conference on Data Mining (ICDM'05). IEEE, Houston. p 8
- Borgwardt KM, Ong CS, Schönauer S, Vishwanathan S, Smola AJ, Kriegel H-P (2005) Protein function prediction via graph kernels. *Bioinformatics* 21(suppl_1):47–56
- Brandes U (2001) A faster algorithm for betweenness centrality. *J Math Sociol* 25(2):163–177
- Bruna J, Zaremba W, Szlam A, LeCun Y (2014) Spectral networks and locally connected networks on graphs. In: International conference on learning representations (ICLR). OpenReview.net, Amherst
- Chiang C-F, Nagaj D, Wocjan P (2010) Efficient Circuits for Quantum Walks. *Quantum Info. Comput.* 10(5):420–434
- Childs AM (2009) Universal computation by quantum walk. *Phys Rev Lett* 102(18):180501
- Debnath AK, Lopez de Compadre RL, Debnath G, Shusterman AJ, Hansch C (1991) Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *J Med Chem* 34(2):786–797
- Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In: Lee D. D., Sugiyama M., Luxburg U. V., Guyon I., Garnett R. (eds). Advances in Neural Information Processing Systems 29. Curran Associates, Inc., Red Hook. pp 3844–3852
- Dernbach S, Mohseni-Kabir A, Pal S, Towsley D (2018) Quantum Walk Neural Networks for Graph-Structured Data. In: Aiello L. M., Cherifi C., Cherifi H., Lambiotte R., Lió P., Rocha L. M. (eds). Complex Networks and Their Applications VII. Springer, Cham. pp 182–193
- Dunjko V, Briegel HJ (2018) Machine learning & artificial intelligence in the quantum domain: a review of recent progress. *Reports on Progress in Physics* 81(7):074001
- Farhi E, Gutmann S (1998) Quantum computation and decision trees. *Phys Rev A* 58(2):915
- Gilmer J, Schoenholz SS, Riley PF, Vinyals O, Dahl GE (2017) Neural Message Passing for Quantum Chemistry. In: Doina P, Yee W. T (eds). Proceedings of the 34th International Conference on Machine Learning. PMLR, Sydney Vol. 70. pp 1263–1272
- Gori M, Monfardini G, Scarselli F (2005) A new model for learning in graph domains. In: Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005. IEEE, Montreal Vol. 2. pp 729–734
- Gupta S, Zia R (2001) Quantum neural networks. *J Comput Syst Sci* 63(3):355–383
- Jing L, Shen Y, Dubček T, Peurifoy J, Skirlo S, LeCun Y, Tegmark M, Soljačić M (2017) Tunable Efficient Unitary Neural Networks (EUNN) and Their Application to RNNs. In: Proceedings of the 34th International Conference on Machine Learning - Volume 70. JMLR.org, Sydney. pp 1733–1741
- Joo J, Knight PL, Pachos JK (2007) Single atom quantum walk with 1d optical superlattices. *J Modern Opt* 54(11):1627–1638
- Jordan SP, Wocjan P (2009) Efficient quantum circuits for arbitrary sparse unitaries. *Phys Rev A* 80(6):062301
- Kendon V (2006) Quantum walks on general graphs. *Int J Quantum Inf* 4(05):791–805
- Kipf TN, Welling M (2016) Semi-Supervised Classification with Graph Convolutional Networks. In: 5th International Conference on Learning Representations, ICLR 2017. OpenReview.net, Amherst
- Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet Classification with Deep Convolutional Neural Networks. In: Pereira F, Burges C. J. C., Bottou L, Weinberger K. Q. (eds). Advances in Neural Information Processing Systems 25. Curran Associates, Inc., Red Hook. pp 1097–1105
- Loke T, Wang J (2011) An efficient quantum circuit analyser on qubits and qudits. *Comput Phys Commun* 182(10):2285–2294
- Loke T, Wang J (2012) Efficient circuit implementation of quantum walks on non-degree-regular graphs. *Phys Rev A* 86(4):042338
- Lovett NB, Cooper S, Everitt M, Trevers M, Kendon V (2010) Universal quantum computation using the discrete-time quantum walk. *Phys Rev A* 81(4):042330
- Manouchehri K, Wang J (2008) Quantum walks in an array of quantum dots. *J Phys A Math Theor* 41(6):065304
- Manouchehri K, Wang J (2009) Quantum random walks without walking. *Phys Rev A* 80(6):060304
- Nayak A, Vishwanath A (2000) Quantum walk on the line. arXiv preprint quant-ph/0010117
- Qiang X, Yang X, Wu J, Zhu X (2012) An enhanced classical approach to graph isomorphism using continuous-time quantum walk. *J Phys A Math Theor* 45(4):045305
- Rohde PP, Schreiber A, Štefaňák M, Jex I, Silberhorn C (2011) Multi-walker discrete time quantum walks on arbitrary graphs, their properties and their photonic implementation. *New J Phys* 13(1):013001
- Rohde PP, Schreiber A, Štefaňák M, Jex I, Gilchrist A, Silberhorn C (2013) Increasing the dimensionality of quantum walks using multiple walkers. *J Comput Syst Sci Nanosci* 10(7):1644–1652
- Rossi MA, Benedetti C, Borrelli M, Maniscalco S, Paris MG (2017) Continuous-time quantum walks on spatially correlated noisy lattices. *Phys Rev A* 96(4):040301

- Rossi L, Torsello A, Hancock ER (2013) A Continuous-Time Quantum Walk Kernel for Unattributed Graphs. In: Kropatsch W. G., Artner N. M., Haxhimusa Y., Jiang X. (eds). *Graph-Based Representations in Pattern Recognition*. Springer, Berlin Vol. 7877. pp 101–110
- Rossi L, Torsello A, Hancock ER (2015) Measuring graph similarity through continuous-time quantum walks and the quantum jensen-shannon divergence. *Phys Rev E* 91(2):022815
- Rupp M, Tkatchenko A, Müller K-R, von Lilienfeld OA (2012) Fast and accurate modeling of molecular atomization energies with machine learning. *Phys Rev Lett* 108:058301
- Ryan CA, Laforest M, Boileau J-C, Laflamme R (2005) Experimental implementation of a discrete-time quantum random walk on an nmr quantum-information processor. *Phys Rev A* 72(6):062317
- Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G (2009) The graph neural network model. *IEEE Trans Neural Netw* 20(1):61–80
- Schomburg I, Chang A, Ebeling C, Gremse M, Heldt C, Huhn G, Schomburg D (2004) Brenda, the enzyme database: updates and major new developments. *Nucleic Acids Res* 32(suppl_1):431–433
- Shenvi N, Kempe J, Whaley KB (2003) Quantum random-walk search algorithm. *Phys Rev A* 67(5):052307
- Shervashidze N, Schweitzer P, Leeuwen EJV, Mehlhorn K, Borgwardt KM (2011) Weisfeiler-lehman graph kernels. *J Mach Learn Res* 12(Sep):2539–2561
- Travaglione BC, Milburn GJ (2002) Implementing the quantum random walk. *Phys Rev A* 65(3):032310
- Velickovic P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y (2017) Graph attention networks. In: *Proceedings of the International Conference on Learning Representations (ICLR)*. ICLR, Amherst
- Vinyals O, Bengio S, Kudlur M (2016) Order Matters: Sequence to sequence for sets. In: *4th International Conference on Learning Representations, ICLR 2016*. OpenReview.net, Amherst
- Wale N, Watson IA, Karypis G (2008) Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowl Inf Syst* 14(3):347–375
- Williams C, Vose R, Easterling D, Menne M (2006) United states historical climatology network daily temperature, precipitation, and snow data ORNL/CDIAC-118, NDP-070. Available on-line [<http://cdiac.ornl.gov/epubs/ndp/ushcn/usa.html>] from the Carbon Dioxide Information Analysis Center, Oak Ridge National Laboratory, USA
- Zhang P, Ren X-F, Zou X-B, Liu B-H, Huang Y-F, Guo G-C (2007) Demonstration of one-dimensional quantum random walks using orbital angular momentum of photons. *Phys Rev A* 75(5):052310

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Submit your manuscript to a SpringerOpen[®] journal and benefit from:

- Convenient online submission
- Rigorous peer review
- Open access: articles freely available online
- High visibility within the field
- Retaining the copyright to your article

Submit your next manuscript at ► springeropen.com
