

PQKELP: Projected Quantum Kernel Embedding based Link Prediction in dynamic networks

Mukesh Kumar^a, Nisha Singh^{b,*}, Bhaskar Biswas^b

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

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

ARTICLE INFO

Dataset link: [Dynamic Dataset \(Reference data\)](#)

Keywords:

Temporal networks
Link prediction
Machine learning
Quantum computing
Projected quantum kernel
Hilbert space

ABSTRACT

In dynamic networks, where the network's topology is constantly changing, link prediction is a challenging issue. The two major challenges in link prediction within a time-varying network are accuracy and efficiency. Although random walk techniques have introduced promising embedding-based approaches, they fall short of optimization. Quantum computing, on the other hand, enhances performance in high-dimensional spaces, yet faces concerns about the limited efficiency of few qubit simulators. Addressing these issues, Projected Quantum Kernels (PQK) presents an elegant solution to achieve quantum advantage by simple quantum projection using the kernel trick, followed by a projection back to the classical state with relabeled data. In this work, we propose Projected Quantum Kernel Embedding based Link Prediction (PQKELP), a projected quantum kernel (PQK) approach on random walk embedding-based features to solve the link prediction problem. Thereby achieving a two-fold improvement by combining embedding generation and quantum projection, resulting in quantum-enhanced embedded features that enhance link prediction performance. Extensive experiments and analyses were done with a set of well-known random walk methods, i.e., Node2Vec, DeepWalk, and Walklets, also with six classical machine learning techniques and on five well-known dynamic network datasets. The results including several performance metrics like accuracy, AUC, F1-score, and precision show that our proposed model is better than traditional link prediction methods, classical machine learning approaches, and even the most cutting-edge methods available currently.

1. Introduction

The link prediction problem stands as one of the fundamental challenges in the realm of complex network research. Link prediction finds extensive applications across diverse disciplines. These applications span various domains, encompassing social networks, drug discovery, research citations, friend recommendations on online social networks, criminal activity detection, knowledge graphs, e-commerce, and anomaly detection, among many others. It can help you recommend new friends on social platforms like Facebook and Instagram (Huang, Li, & Chen, 2005), automatic hyperlink construction (Adafre & de Rijke, 2005), recommending future citations in a citation network (Dileep & Deepthi L., 2022). In addition, link prediction is being utilized to develop recommendation systems for e-commerce websites that generate new item suggestions for users (Kumar Sharma, Bajpai, Adhvaryu, Dhruvi Pankajkumar, Parthkumar Gordhanbhai, & Kumar, 2023), for predicting protein-protein interactions (PPI) and bio-informatics (Airoldi, Blei, Fienberg, Xing, & Jaakkola, 2006), to reveal hidden

relationships between terrorists and their organizations (Muniz, Goldschmidt, & Choren, 2018), in the field of security.

Studies have incorporated a variety of link prediction perspectives for various types of graphs. The basic framework for link prediction approaches revolves around similarity-based, probabilistic, domain/semantic similarity-based, stochastic block models, and matrix factorization models (Isah, Neagu, & Trundle, 2015; Kumar, Singh, Singh, & Biswas, 2020). The fact that networks change over time further complicates the task. To make predictions about links in future networks, patterns in past network structures must be looked at in temporal networks Fig. 1. Our work mainly concentrates on similarity-based methods that use node neighborhoods and network structure to compute the similarity between the nodes in the network. These are local, quasi-local, or global similarity-based methods (Liu, Li, Fiumara, & De Meo, 2023).

A random walk-based link prediction task takes advantage of walks that can be extracted from graph structure and uses this data to create better features (Huang, Silva, & Singh, 2021) It leads to the extraction

* Corresponding author.

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

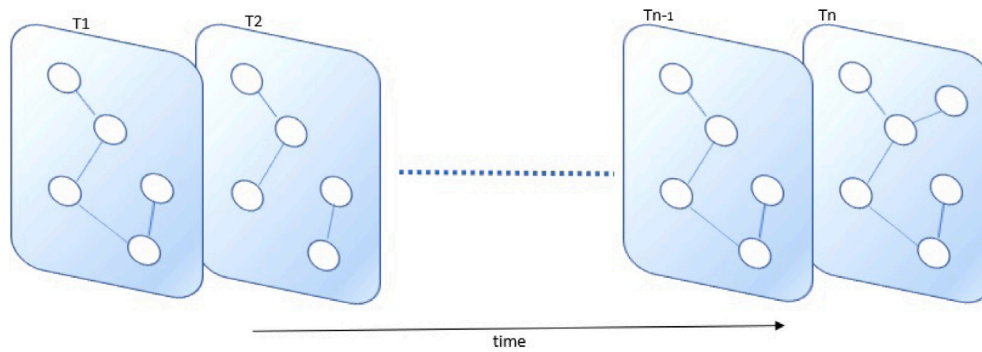


Fig. 1. Dynamic network with different time snapshots.

of random walks that are utilized for generating node embeddings with the help of skip-gram techniques, very similar to the Word2Vec techniques used in natural language processing. Based on this idea, Perozzi, Al-Rfou, and Skiena (2014) came up with a Deepwalk method to think of network paths (walks) like words in a sentence and create embedding for network nodes instead of words. In the year 2016, Grover et al. proposed a novel approach called Node2Vec, a biased random walk that explores diverse neighborhoods while maintaining local neighborhood prioritization (Grover & Leskovec, 2016). Subsequently, in Perozzi, Kulkarni, Chen, and Skiena (2017a), the approach was to use walklets that generate multi-scale relationships by subsampling short random walks on the vertices of a graph. These walks capture successively higher-order relationships from the adjacency matrix.

Quantum computing (Han & Kim, 2002; Narayanan & Moore, 1996; Shor, 1994) is a new branch of computer science based on quantum mechanics (Ganguly, 2021). Since its inception at the turn of the twentieth century, quantum theory has been the most comprehensive description of small-scale physics. Quantum machine learning (QML) is a multidisciplinary field that combines two of the most fascinating fields of research: quantum computing and classical machine learning. Quantum computing and machine learning (QML) are undoubtedly two of science's and industry's "hottest" areas. Parallel to recent developments in deep learning, interest in quantum computing has grown significantly in academia and industry (Preskill, 2018). Several attempts have been made to improvise link predictions with QML. In Liu et al. (2022), the authors proposed the utilization of hybrid quantum-classical neural networks. The objective here is to enhance embedding for more effective link prediction.

Here, we attempted to utilize quantum enhancements to optimize embedded features generated by random walks. These features are embedding vectors generated with a random walk from each network node. There has been extensive research in the areas of random walk-based link prediction and quantum optimization-based link prediction. In previous attempts to explore the impact of quantum machine learning with random walks, researchers devised quantum random walks that incorporate quantum computation and enhancements into random walks (Kadian, Garhwal, & Kumar, 2021). In their paper (Zhiwei, Yichao, Jihong, & Shuigeng, 2018) compared random walks to ant colonial walks and used pheromones to change the probabilities of the random walks utilizing the quantum walk's spreading behavior that spreads information far more quickly than the classical random walk (Magniez, Nayak, Richter, & Santha, 2012; Orthey & Amorim, 2017).

Instead of the earlier approach of using quantum techniques for modifying random walks, we suggest using quantum enhancements to improve embedding-based features. These features are produced after random walks are generated and embeddings are calculated. We wish to leverage quantum advantages such as high-dimensional space, entanglement, and superposition into the subsequent task of link prediction by applying quantum projection to these created embeddings. Manipulation and classification of huge quantities of vectors

in high-dimensional spaces are common issues in machine-learning tasks. Solving such issues using traditional algorithms usually takes a time polynomial in the number of vectors and the dimension of the space. Quantum computers excel at processing massive tensor product spaces and high-dimensional vectors (Huynh, Hong, Mian, Suzuki, Wu, & Camtepe, 2023). In both the number of vectors and their dimension, quantum machine learning can take logarithmic time, which is an exponential speedup over classical techniques.

The following are the primary motivations for developing this approach:

- Quantum machine learning tends to combine the substantial performance boost provided by quantum paralleling and enabling machine learning algorithms to learn high-dimensional data, making it an emerging and promising research area (Huang, Broughton, et al., 2021).
- Quantum models have a very similar mathematical structure to kernel methods, i.e., they analyze data in high-dimensional Hilbert spaces (Schuld, 2021). Projecting back the data from quantum space to classical space significantly improves the performance (Huang, Broughton, et al., 2021).
- Random walks provide enhanced features in the form of embeddings; hence, the performance of overall link prediction turns out to be better than providing raw topological features for link prediction.
- Furthermore, significant progress has been made in the realm of random walk techniques. These advancements aim to enhance traditional word vector-based embeddings by incorporating biased random walks in diverse ways. The result is more accurate and context-aware network representations.
- Previous attempts to merge quantum computing and random walk methodologies for link prediction primarily revolved around leveraging quantum enhancements within biased random walks. To the best of our knowledge, no known research or work has explored the utilization of random walk-enhanced features specifically for quantum projection tasks.

The main contributions of this work are as follows:

- Our research is built on the notion of creating a quantum-assisted feature-based novel approach for link prediction with the integration of Projected Quantum kernel (PQK) and random walk embeddings with machine learning models to accelerate link prediction performance.
- In our work for link prediction, we have used enhanced features not only improvised by embeddings generated from random walk models but also by projecting the data to quantum high dimensional space.
- We have used a kernel for quantum enhancements for projecting data to quantum space and then back to classical state, hence the algorithm is efficient on both real machines as well as simulators, unlike pure quantum classifiers that take a lot of time on simulators.

- We have done intensive results analysis with three different random walk-based models: Node2Vec, Walklet, and Deepwalk on a set of five test ratios. The analysis is presented for several quantum-enhanced machine learning models like neural Network (QE-NN), XGBoost (QE-XGB), Logistic Regression (QE-LR), Random forest classifier (QE-RFC), Linear discriminant Analysis (QE-LDA) and Gaussian Naive Bayes (QE-GNB). Subsequently, combined features from all random walk techniques are projected with quantum kernel yielding PQKE-* model and compared with all the said machine learning models and the state-of-the-art methods.
- Extensive experiments on five well-known dynamic datasets with four performance metrics show that our proposed *PQKELP* model i.e. Projected Quantum Kernel on Quantum Random Walk Embedding features reveals better results.

Our idea to incorporate three promising domains, namely quantum enhancement, projection with the kernel, and random walk-based embedded features for link prediction was drafted with inspiration from previous works such as Huang, Broughton, et al. (2021) and Abadi et al. at TensorFlow (Abadi, Agarwal, Barham, Brevdo, Chen, Citro, Corrado, Davis, Dean, Devin, Ghemawat, Goodfellow, Harp, Irving, Isard, Jia, Jozefowicz, Kaiser, Kudlur, Levenberg, Mané, Monga, Moore, Murray, Olah, Schuster, Shlens, Steiner, Sutskever, Talwar, Tucker, Vanhoucke, Vasudevan, Viégas, Vinyals, Warden, Wattenberg, Wicke, Yu, & Zheng, 2015). After conducting extensive experimentation with variations from ML techniques, random walks, test ratios, performance metrics, and datasets, we established the superiority of the proposed approach.

Organization The rest of the paper is organized as follows. Section 2 contains preliminaries for our approach, Section 3 details the related work in the field, Section 4 contains the proposed approach, Section 5 contains results analysis and Sections 6 and 7 presents conclusion, limitation and future work.

2. Preliminaries

The intricate concepts of this work include understating the dynamic networks that evolve over time, exiting nodes and links may vanish and new nodes and links may arise. In such a complex network the feature extraction spans several snapshots and this becomes more complex when we target extracting embedded features. To do this an in-depth understating of random walk-based embedding is essential. Next, it is crucial to understand the quantum computation basics to realize the quantum enhancements achieved. A brief introduction of these concepts from the vast information collected is presented here.

2.1. Dynamic networks

A dynamic network or graph can be used to depict an ordered set of timed occurrences. It comprises deletions and additions to nodes and edges. When someone joins a social network like Facebook or Twitter, a new node is created. When they follow another user, a link is established. When they change their profile, the node is updated. Link prediction is the technique of detecting new or missing links in a dynamic network at discrete timestamps as shown in Fig. 1. Consider a dynamic network, denoted as $G = (V, E)$ where V represents the set of vertices, and each edge $(u, v) \in E$, symbolizes the connection or relationship between u and v . Let $G_1, G_2, G_3, \dots, G_n$ be networks observed at different time points $t_1, t_2, t_3, \dots, t_n$. Our task is to predict the state of G at time $t + 1$, denoted as G_{t+1} . Dynamic networks offer a more intricate representation of complex relationships in comparison to static networks. Nodes represent entities, and links represent relationships, with each link containing temporal information and other relevant attributes.

Dynamic network analysis finds diverse applications across a variety of domains. These dynamic networks are characterized by the appearance and disappearance of nodes and edges over time where each time interval is represented as a discrete static network (Casteigts, Flocchini, Quattrociochi, & Santoro, 2012; Divakaran & Mohan, 2019; Kumar, Mishra, & Biswas, 2021). In the context of dynamic networks, Sarkar, Chakrabarti, and Jordan (2012) introduced a non-parametric method for predicting links in temporal networks. This method involves dividing the time dimension into sub-sequences of graph snapshots to make predictions regarding temporal network links. The topological properties and local neighbors are used in this strategy. Fig. 1 shows a dynamic network with different snapshots.

2.2. Classical machine learning methods

Some machine learning approaches can be used to determine a node pair's missing or future link in a machine learning approach (Kumar et al., 2021; Martínez, Berzal, & Cubero, 2016; Pecli, Cavalcanti, & Goldschmidt, 2018; Wang, Xu, Wu, & Zhou, 2015). In the context of the link prediction problem, machine learning algorithms approach the prediction process as a classification task. The classifier is responsible for making predictions about the class i.e. whether an edge exists. Alternative methods, such as similarity and probabilistic techniques, employ similarity or probabilistic functions to compute a score for each unobserved link (Das & Das, 2017). On the other hand, the link prediction challenge can also be addressed by utilizing a learning-based model that integrates topological graph properties and attribute data (Zhou, Kong, Zhang, Liu, & Jin, 2020). This approach combines graph structure and feature information for making predictions.

In binary classification problem a number of classifiers can be employed to predict the label of unknown data points, including Neural Networks, the boosting technique XGBoost, Logistic regression, Random Forest Classifiers, Linear Discriminant Analysis, Gaussian Naive Bayes (Al Hasan & Zaki, 2011; Kumar et al., 2020; Liu, Jiang, Wang, Xie, & Ni, 2020). Creating an acceptable feature set is one of the model's primary challenges. Some recent techniques employ random walk for generating node/edge embedding for link prediction. They are based on Word2Vec algorithm following principal of Markov Chain rule. These techniques have proven to be extremely successful when combined with the traditional classifiers (Amara, Taieb, & Aouicha, 2021).

2.3. Quantum computation

Quantum computing is a phenomenon that applies quantum mechanics physical principles to computational tasks in computer science (Hirvensalo, 2003) for e.g. Factorization (Shor, 1994), searching (Grover, 1996), Cryptography (Brassard, 1994), etc. This is done to leverage the advantage of high-dimensional Hilbert space with data projection. This approach is distinguished by an extension of probability theory that enables unique phenomena such as superposition and entanglement, which are challenging to replicate using conventional computers (Feynman, 2018). One key discovery that has led to the application of quantum computing in machine learning is their ability to do fast linear algebra on a state space that grows exponentially with the number of qubits. Researchers have demonstrated that quantum computation can simulate problems that traditional computers cannot and can achieve quadratic or exceptional speed in some cases. With the development of quantum machine learning, quantum computing emerges as a subject with promise for machine learning, an area of the current study that is expanding and showing promise. In this section, we introduce the basic jargon of Quantum Computing like Qubits, Quantum gates, and Quantum states and quantum phenomena like high-dimensional Hilbert Space, entanglement, superposition, and quantum interference.

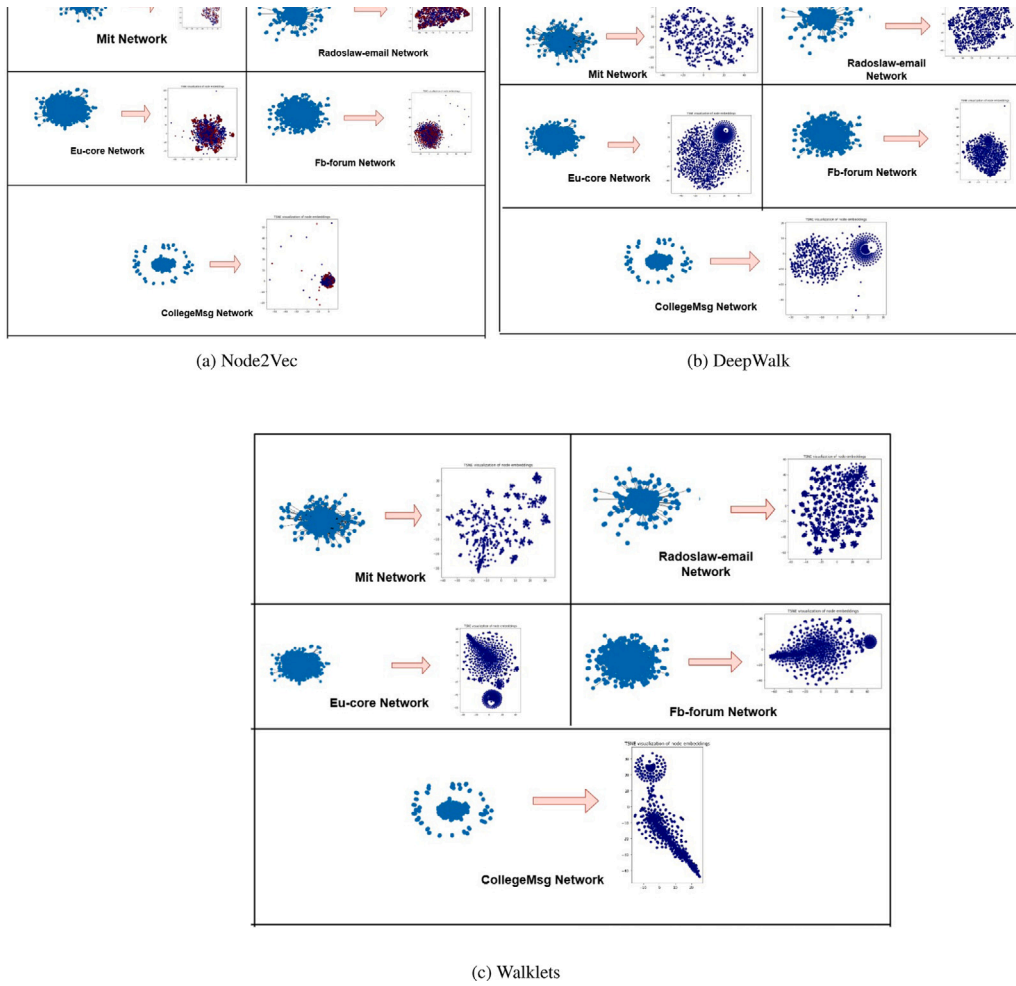


Fig. 2. Analysis of different random walk Embedding algorithms (Node2Vec, DeepWalk and Walklet) with its different node embedding algorithms on different dynamic networks.

- **The Qubit:** Like a classical bit, a quantum bit, also known as a qubit, is the smallest information unit held in a two-state quantum computer (Benioff, 1980; Ganguly, 2021; Han & Kim, 2002). Unlike a classical bit having two constant states “0” and “1”, a quantum bit can also exist in superposition states, this is caused by incompatible measurements or entanglement with other quantum bits. The key difference and superior power of qubits over classical bits lies in their capacity to harness the powers of superposition, interference, and entanglement.
- **The state of a Qubit:** The state of a qubit is represented by $|\psi(t)\rangle$ which refers to a vector that is an element of abstract complex vector spaces basically the Hilbert Space. Mathematically, the state of a qubit, ψ is represented as.

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \quad (1)$$

α and β represent the probability amplitudes of the states 0 and 1 respectively. They necessarily satisfy the following conditions.

$$\alpha^2 + \beta^2 = 1 \quad (2)$$

This may also be visualized geometrically on the Bloch sphere as in Fig. 4

- **The Hilbert Space:** One particular type of linear vector space is the Hilbert Space (H). It possesses several extra traits in addition to all the characteristics of linear vector space. An inner product operation in Hilbert Space represents the separation between two vectors. One definition of this inner product is a $\langle\psi_1, \psi_2\rangle \in$ set of complex numbers. Additionally, it grants the two-state

system's components the high dimensional complex vector space advantage.

- **Quantum Gates:** Quantum gates are circuits that perform different coherent quantum operations on quantum data, such as qubits, resulting in the qubit's amplitude or phase change or qubits being entangled. Gates are unitary operations and because they are applied on vectors they can be represented as unitary matrices (Knill et al., 2008). It can be visualized as rotations around the Bloch sphere as in Fig. 4. The gates that perform these operation on the qubit can be single-qubit gates for e.g. Pauli Gates, Hadamard Gate, Phase Gate etc. or two-qubit gates like CNOT Gate, SWAP Gate, Controlled Phase Shift Gate etc. or even three-qubit gates like Toffoli Gate, Fredkin Gate etc.
- **Quantum Enhancement:** We generally hear about the possibility of quantum computers generating computations that are hard to simulate classically, the interesting fact is, that it is due to the spooky concept of quantum entangled. Two entangled qubits will be dependent on each other which is taken from the phenomenon of Quantum Mechanics where changing state of one ion changes the state of another even when they are separated far apart. Mathematically any state that cannot be expressed as a tensor product of an individual state is considered to be entangled, for example:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \quad (3)$$

Superposition, another well-known idea in quantum computing, holds that a qubit can be in two states of superposition simultaneously; this allows quantum computers to perform exponentially

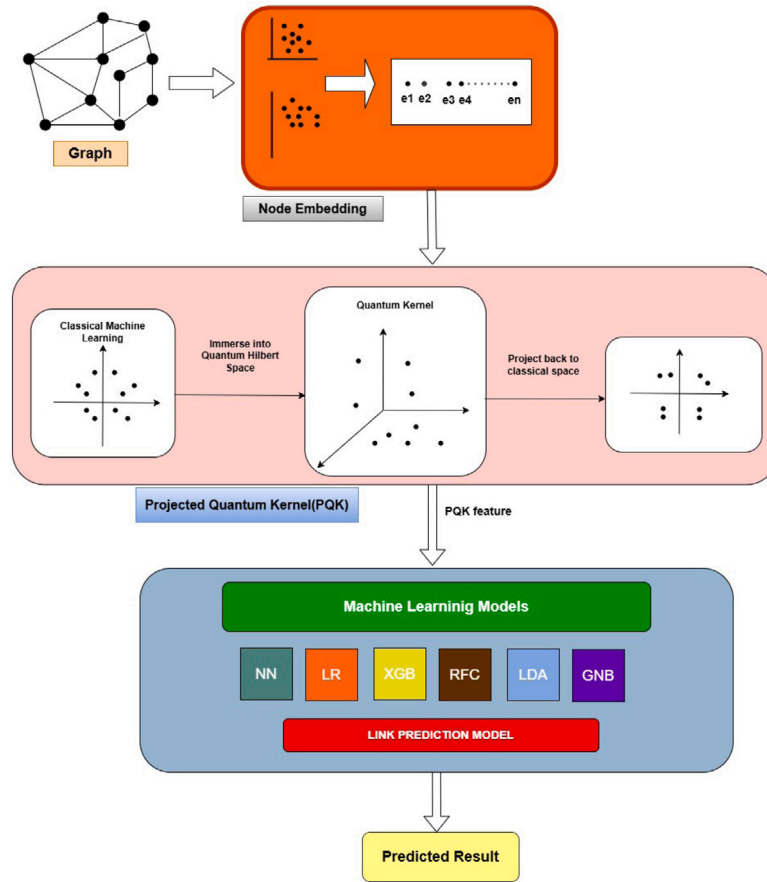


Fig. 3. Proposed Model Framework.

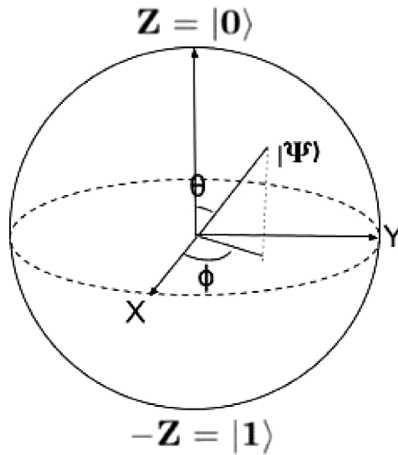


Fig. 4. Geometric Visualization of qubit state on Bloch Sphere.

faster than classical computers (Van Gael, 2005). Another benefit that we get out of quantum computation is a better representation of probabilities into high-dimensional Hilbert space instead of linear probabilities which results in better accuracy and predictions (Ismael, 2021). Quantum interference a dual phenomenon of entanglement is nothing but the capability of a qubit to interfere with the state of another qubit and disrupt each other. This results in the prevention of repeatable processes and may lead to finding better and faster paths.

2.4. Quantum machine learning

Machine learning and quantum computing are two technologies that have the potential to change the way computation is conducted in order to solve challenges that were previously unsolvable. Quantum machine learning is a term coined to describe the convergence of machine learning and quantum computing, and it has drawn a lot of exposure in recent years (Arunachalam & de Wolf, 2017; Ciliberto et al., 2018; Dunjko & Briegel, 2018). As a result, several quantum algorithms have recently been presented (Biamonte et al., 2017; Farhi & Neven, 2018; Mitarai, Negoro, Kitagawa, & Fujii, 2018; Romero, Olson, & Aspuru-Guzik, 2017; Wan, Dahlsten, Kristjánsson, Gardner, & Kim, 2017). Lloyd, Mohseni, and Rebentrost (2013) have proposed Quantum algorithms for supervised and unsupervised machine learning. The use of quantum information processors to conduct machine learning tasks is investigated in several results on quantum machine learning. It includes pattern matching (Sasaki & Carlini, 2002), Probably Approximately Correct (PAC) learning (Atıcı & Seredidio, 2007), binary classifiers (Pudenz & Lidar, 2013), and quantum support vector machines (Anguita, Ridella, Riveccio, & Zunino, 2003).

QML-Domains: Recent advances in quantum enhanced machine learning (QML) investigate how quantum computing may outperform traditional techniques, particularly when dealing with complicated and large-scale datasets. The superposition and entanglement properties of qubits are used by quantum models, such as Quantum Neural Networks (QNNs) and Variational Quantum Classifiers (VQCs), to analyze and categorize high-dimensional data more quickly than conventional neural networks. For example, optimization problems, drug discovery, material science, climate modeling and simulation, pattern recognition, Cryptography and security, Financial Modeling and risk assessment. Quantum Enhancement Optimization (QEO) (Jiang, Rieffel, & Wang,

2017) is one important area of research that aims to improve optimization algorithm computations using quantum enhancements. By embedding properties into qubits, QEEGNet (Chen, Chen, Tsai, & Wei, 2024), incorporates quantum layers into neural networks to classify EEG data more effectively. When compared to purely classical models, it exhibits greater accuracy on standard datasets. The study in Rehman (2024) proposes an advanced image encryption system that combines chaotic maps and quantum cryptographic principles to enhance data security.

- **QML Algorithms:** Researchers are developing quantum counterparts to classical machine learning algorithms. The first generation of quantum machine learning (QML) algorithms, including principal component analysis (Lloyd, Mohseni, & Rebentrost, 2014), support vector machine (Rebentrost, Mohseni, & Lloyd, 2014), K-means clustering (Lloyd et al., 2013), and recommendation systems (Kerenidis & Prakash, 2016). QML employs entanglement through CNOT gates and superposition through Hadamard gates to project classical data to quantum space (Terashi, Kaneda, Kishimoto, Saito, Sawada, & Tanaka, 2021). The data is then trained using quantum optimization and parameter shift techniques 5. Last but not least, observable is used in quantum measurements to translate quantum probabilities into classical outcomes (Yano, Suzuki, Itoh, Raymond, & Yamamoto, 2021).
- **QML Explanation:** In Havlíček, Córcoles, Temme, Harrow, Kandala, Chow, and Gambetta (2019), the authors employed the following parameterized circuit, used to encode data where the classical data will be first projected to complex quantum vector space using a parameterized feature map circuit, mathematically expressed by $U_{\phi}(x)$ as

$$U_{\phi}(x) = \prod_d \bigcup_{\phi} (x) H^{\otimes n} \quad (4)$$

Where d is the depth of the circuit i.e. number of repetitions of Hadamard gate H and entangling block $U_{\phi(x)}$

$$U_{\phi(x)} = \exp \left(i \sum_{x \subseteq [n]} \phi_s(x) \prod_{k \in s} P_i \right) \quad (5)$$

$P_i \in \{I, X, Y, Z\}$ denotes the Pauli matrices, S describes the connectivity between qubits or data points $S = \left\{ \binom{n}{k} \right\}$, $k=1, \dots, n$, $\phi_s(x)$ is the data mapping function. As shown in Fig. 5 the H (Hadamard) gate projects the vectors to $|+\rangle$ and $|-\rangle$. They serve the purpose of complex orthonormal basis and $U_{\phi(x)}$ performs entanglement and rotation along the Z axis, which makes the circuit hard to simulate classically (Havlíček et al., 2019).

Then in the training phase, classifiers in quantum machine learning basically encapsulate these parameterized circuits and use optimization techniques to quantify and eventually classify the circuit outcome. Parameterized quantum circuits, where the gates are defined through tunable parameters, are a fundamental building block of quantum machine learning algorithms.

2.5. Projected quantum kernel (PQK) technical discussion

A kernel function signifies nonlinear feature mapping $\phi(x)$ that maps x to infinite-dimensional feature space, such that the difference between two such vectors may be captured through $k(x_i, x_j) = \phi(x_i) \phi(x_j)$. This forms the foundation of the well-known 'kernel trick,' which allows the implementation of complex and powerful mappings represented by $\phi(x)$ through the evaluation of relatively straightforward kernel functions denoted as k . For kernel-based Machine Learning models, the training function can be written as $h(x) = w^\dagger \phi(x)$ where w is a vector in the kernel-defined feature space. For example, a convolutional neural network with many hidden layers can be easily trained with corresponding neural tangent kernel k_{CNN} (Jacot, Gabriel, & Hongler,

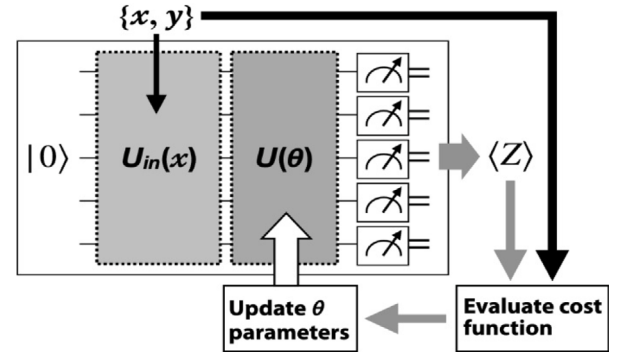


Fig. 5. Phases of Variational Quantum Algorithm.

2018). The feature map ϕ_{CNN} is a nonlinear mapping that extracts all local properties of x for the kernel k_{CNN} . On a broader level, a kernel function can be thought of as a measure of similarity, as for any pair of values, $k(x_i, x_j)$ is large when x_i and x_j are close. A kernel function may be represented as a matrix $K_{ij} = k(x_i, x_j)$ when considering finite input data.

A wide class of functions can be learned with a sufficiently large amount of data by using the right kernel function k . For a simple quantum projection, a kernel $k(x_i, x_j) = |\langle x_i | x_j \rangle|$ corresponds to a feature map

$$\phi(x) = \sum_{kl} x_i^k * x_j^l |k\rangle \otimes |l\rangle \quad (6)$$

which has the capacity to learn quadratic functions in the amplitudes. For example the quantum kernel

$$k^Q(x_i, x_j) = |\langle x_i | x_j \rangle| = \text{Tr}(\rho(x_i) \rho(x_j)) \quad (7)$$

can learn arbitrarily deep quantum neural network DQNN and the Gaussian kernel,

$$k^y(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2) \quad (8)$$

with hyper-parameter γ , can learn any continuous function in a compact space, which includes learning any QNN. According to Huang, Broughton, et al. (2021), the performance of the methods can be considerably improved by projecting back from the quantum space to a classical one in the projected quantum kernel model. Below are some simple choices based on reduced density matrices (RDMs) of the quantum state.

• Linear kernel function with 1-RDMs

$$Q_i^1(x_i, x_j) = \sum_k \text{Tr}[\text{Tr}_{m \neq k}[\rho(x_i)] \text{Tr}_{n \neq k}[\rho(x_j)]] \quad (9)$$

where $\text{Tr}_{m \neq k}(\rho)$ represents a partial trace of the quantum state ρ over all qubits except the k th qubit. Any observable that can be expressed as a sum of one body terms could be learned by it.

• Gaussian kernel function with 1-RDMs

$$Q_s^1(x_i, x_j) = \exp(-\gamma \sum_k (\text{Tr}_{m \neq k}[\rho(x_i)] - \text{Tr}_{n \neq k}[\rho(x_j)])^2) \quad (10)$$

where γ represents here as a hyper-parameter. Any nonlinear function of the 1-RDMs could be learned by it.

• Linear kernel with k-RDMs

$$Q_i^k(x_i, x_j) = \sum_{K \in S_k(n)} \text{Tr}[\text{Tr}_{n \notin K}[\rho(x_i)] \text{Tr}_{m \notin K}[\rho(x_j)]] \quad (11)$$

where $S_k(n)$ represents the set of subsets of k qubits from n , and $\text{Tr}_{n \notin K}$ denotes a partial trace of qubits not in subset K . Any observable that can be expressed as a sum of k -body terms could be learned.

2.6. Projected quantum kernel (PQK) algorithm used in the proposed work

The Quantum algorithms used in this article will be covered in this part including all phases right from quantum feature projection to projection back to relabeled classical feature. We used the open source code available on TensorFlow for quantum feature enhancement (Abadi et al., 2015). The basic code for quantum data is available as open source repository.¹ The algorithm begins with the extraction of a feature set obtained from topological matrices that capture the structural properties of the network. In the quantum feature generation, various algorithms have been taken into consideration such as an algorithm for producing the wall of single qubit rotations, creation of a circuit that produces $V(\theta)$, kernel matrix computation, to generate the kernel data points eigenvalues and eigenvectors and to generate new labels that maximize the geometric distance between kernels. Thus algorithm produces features that are relabeled based on quantum optimization. Various steps involved in feature generation by Projected Quantum Kernel (PQK) are shown in Fig. 6. The major components of the algorithm as mentioned in Abadi et al. (2015) are covered here in brief:

1. **Projection through rotation with Single qubit wall:** In this step the task is to create a wall of 1-RDM on all qubits as:

$$V(x_{train}/n_{tritter})^{n_{tritter}} \cup_{1qb} |0\rangle \quad (12)$$

We basically apply single qubit random rotations one by one to each of the qubits. The algorithm iterates over each qubit and over each quantum gate basically the set of Pauli [X,Y,Z] gates, and applies random rotations to each of the gates. We may visualize this as all the qubits pass through each of the Pauli X, Y, and Z gates to which random rotations are applied.

2. **Creating $V(\theta)$, the parameterized circuit:** Then create a parameterized circuit in order to make it hard to simulate classically.

$$V(\theta) = e^{-i \sum_i \theta_i (X_i X_{i+1} + Y_i Y_{i+1} + Z_i Z_{i+1})} \quad (13)$$

The algorithm iterates over each pair of qubits and applies Pauli X, Y, and Z rotations to i th and $i + 1$ th qubit.

3. **Computation for kernel matrix:** Next task is to compute the kernel matrix as discussed earlier the Gaussian Kernel method is used to compute the kernel values for each pair of qubits. In order to calculate expression in Eq. (10) the algorithm first computes the scaled gamma with a product of standard deviation and projected quantum features. This gives us the PQK kernel with the Gaussian kernel approach.
4. **Generation of eigenvalues and eigenvectors for the kernel:** Further it finds the spectrum of the PQK kernel by finding eigenvalue and eigenvectors of the kernel datapoints. The process is done for both projected PQK features as well as classical features.
5. **Production of stilted dataset:** The objective here is the Generation of new labels that maximize the geometric distance between the two kernels. The algorithm here first finds the scaling matrix by using both the classical and PQK eigenvalues and eigenvectors and then computes the eigenvector of the resultant matrix. Lastly, relabel the classical feature by scaling its eigenvector with the largest eigenvector of the resultant matrix.

3. Related work

The advancements in link prediction proceeded in several directions traversing the paths of traditional machine learning methods involving features from several categories like structural similarity-based, node characteristics domain-based, probabilistic or stochastic, etc. (Ahmed

et al., 2018; Ma, Sun, & Qin, 2017; Yasami & Safaei, 2018). Classical approaches to prediction for dynamic with Machine learning have been attempted by Fire et al. (2011), David, Ryan, and Rossi (2012) and Fu, Song, and Xing (2009) where the attempt is made to capture topological similarity based features and enhance the prediction performance with several machine learning techniques.

Bliss, Frank, Danforth, and Dodds (2014) proposed a way to capture not only network but also node characteristics into consideration by employing 16 different classified features from both categories. The Covariance Matrix Adaptation Evolution Strategy (CMAES) is further enhanced with the weights resulting in improved performance. Clauset, Newman, and Moore (2004) does cluster analysis with a statistical technique called the hierarchical model where the network nodes are represented by dendrograms having leaf nodes and internal nodes. The internal nodes are associated with probability representing the weight of the relationship between its descendant nodes. Geyer (1992) employed a Markov chain Monte Carlo technique to generate a collection of dendrograms, with their likelihood determining the probability of selection. Stochastic block models are community-based link prediction methods where the assumption is that the network is divided into communities/blocks then the probability of a link with maximum likelihood can be computed with Bayes theorem (Guimerà & Sales-Pardo, 2009). Zhu, Guo, Yin, Steeg, and Galstyan (2016) have attempted to add a temporal regularize into the matrix factorization framework in order to construct a temporal latent space from snapshot networks. Machine learning can be applied in a variety of industrial applications, such as the mechanical and health sectors, to improve prediction and accuracy (Feng, Ji, Zhang, Ni, Liu, & Beer, 2023; Ni, Ji, Feng, Zhang, Lin, & Zheng, 2024).

3.1. Embedding-based link prediction

Recently, random walks have received much attention in learning network embeddings, in particular, to generate context vertices (Grover & Leskovec, 2016; Perozzi et al., 2014). Consider a random walk of length L that begins at a vertex v_0 of the input graph G and ends at a vertex v_L . If we are at vertex v_t at time t , we will travel to a neighbor of v_t with probability $\frac{1}{d_{v_t}}$ at time $t+1$. As a result, the ensuing sequence of vertex indices ($v_t : t = 0, 1, \dots, L-1$) is a Markov chain. The network embedding approach is a dimensionality reduction technique that preserves the node neighborhood structures while mapping higher-dimensional nodes (vertices) in graphs to a lower ($d \ll D$) dimensional representation (embedding) space. To put it another way, identify the embedding of nodes to lower d -dimensions so that similar nodes (from the original network) have similar embeddings (in the representation space).

There are two types of unsupervised network embedding algorithms currently available (Liu, Zhang, Lü, & Zhou, 2011a): structure-preserving network embedding methods, which only use network structure to learn node embeddings, and attributed network embedding methods, which combine node attributes along with network structure to give a better network embedding.

3.2. Structure preserving network embedding

In structure-preserving network embeddings (Yan et al., 2006), node linkages are vectorized on their own, without using extra information from node characteristics (Shaw & Jebara, 2009; Yan et al., 2006). Embedding can be based on nodes or edges or hybrid embedding, combining node + edges or nodes + community (Cai, Zheng, & Chang, 2018). Taking inspiration from Skip-Gram (Mikolov, Chen, Corrado, & Dean, 2013), DeepWalk (Perozzi et al., 2014) extracts local information from a shortened random walk and embeds the nodes in representation space by considering the walk as a sentence in the language model (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013; Mikolov, Sutskever, et al., 2013). Nove2vec (Grover & Leskovec, 2016) improves

¹ https://www.tensorflow.org/quantum/tutorials/quantum_data/

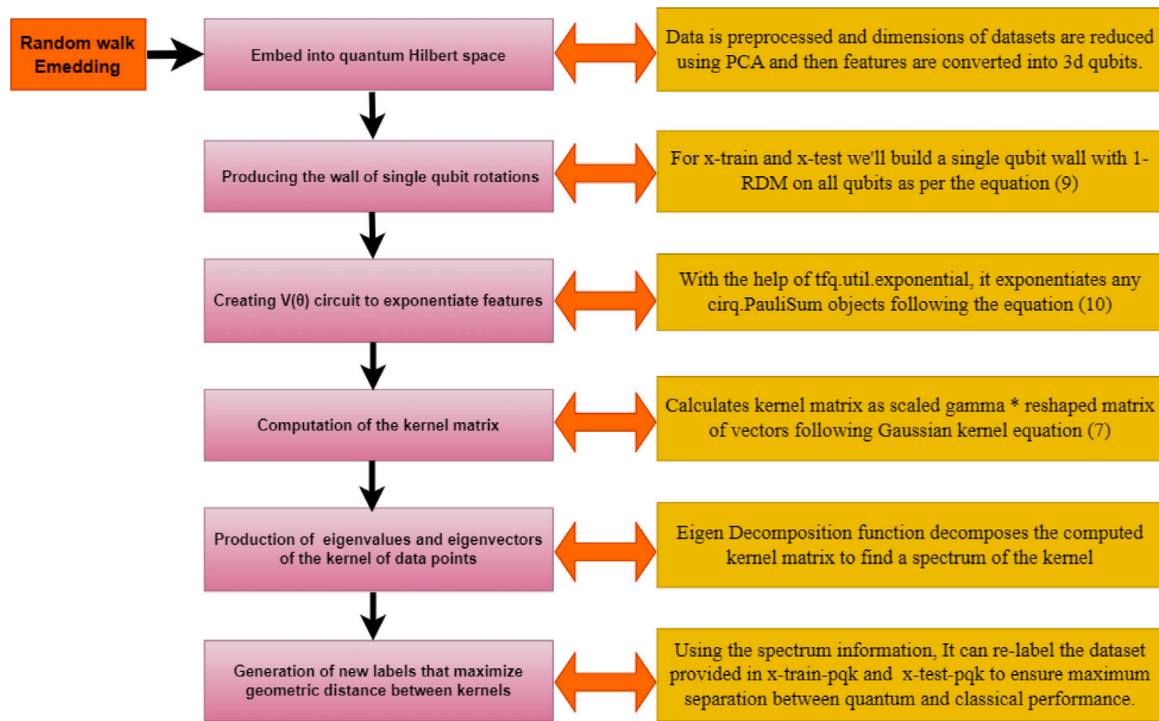


Fig. 6. Steps for Feature Generation in Projected Quantum Kernel (PQK).

on DeepWalk by creating a biased random walk that enhances the possibility of learning the network embedding from flexible network neighbors. It employs a random walk to retain higher-order proximity; however, it is biased, which is a trade-off between BFS and DFS (DFS).

The results of the experiments reveal that Node2vec outperforms Deepwalk. LINE (Tang et al., 2015) models the first-order and second-order adjacency for each pair of nodes from the standpoint of a joint probability distribution. By modeling the relationships between nodes and their k-step neighbors, GraRep (Cao, Lu, & Xu, 2015) goes a step further to address high-order proximities. MNMF (Wang et al., 2017) uses both a proximity matrix and a community membership indication matrix based on NMF to acquire knowledge about network embedding. To acquire knowledge about node representations that retain first-order and second-order closeness, SDNE (Wang, Tang, Morstatter, & Liu, 2016) creates a semi-supervised autoencoder. These algorithms rely solely on network topology, disregarding node content information. These techniques fail to deliver adequate results in cases when the linkages are sparse. The network embedding methods listed above presume that networks are complete.

3.3. Attributed network embedding

To deal with network sparsity, the idea of attributed network embedding (Chang et al., 2015) was created. The goal was to improve performance by adding domain information to the network through node characteristics. Through inductive matrix factorization (Natarajan & Dhillon, 2014), TADW adds node text properties to network embedding. By requiring first-order closeness in the embedding space, HSCA (Zhang, Yin, Zhu, & Zhang, 2016) improves TADW. PRBM (Wang et al., 2016) uses the Restricted Boltzmann Machine (Hinton & Salakhutdinov, 2006) to create node representations using node characteristics while still preserving the similarity between linked nodes. Through neighborhood feature aggregation, GraphSAGE (Hamilton, Ying, & Leskovec, 2017) infers node representations inductively from node content characteristics. NetPLSA (Mei, Cai, Zhang, & Zhai, 2008) looks at both the topology of the network

and the text data to model topics. The topic distributions of nodes can be thought of as embedding representations.

Miao et al. in Xu, Jin, and Zhou (2013), revealed that matrix completion tasks may be speeded up to the order of linear instead of quadratic. Cheng et al. Yang, Liu, Zhao, Sun, and Chang (2015) developed a semi-supervised deep learning model that leverages a random surfing approach to capture the overall structure while also integrating the textual characteristics of vertices using the PV-DBOW model to acquire dependable low-dimensional vector representations. Zhu, Ye, Zhao, and Zhang (2018), introduces the concept of Community and Text-Enhanced DeepWalk (CTDW) by utilizing an inductive matrix completion algorithm. CTDW integrates community attributes and textual information from vertices within the context of matrix factorization, enhancing the Node Representation Learning (NRL) process.

When networks have missing data, especially on node properties, their performance suffers drastically as a result of their inability to manage missing data. Furthermore, because most of these systems rely on matrix factorization and deep learning techniques, they have limited scalability. Our research, on the other hand, presents a simple and successful method for learning node embeddings that are resilient to missing data.

3.4. Quantum random walks

Another related stream of link prediction techniques is quantum random walks, which are the quantum counterparts of classical random walks. In a classical random walk, randomness arises from the random transition between states for a walker that has occupied a definite state. However, in a quantum walk, randomness arises from the quantum superposition of states, non-random, reversible unitary evolution, and collapse of the wave function due to state measurements.

We know that random walks-based graph embedding approaches are very popular for link prediction due to their integrated property of random graph exploration and node representation in terms of embeddings. These embeddings form the enhanced set of features for link prediction. They have three essential components: random walk

Algorithm 1: Algorithm for Quantum Embedding(QE) Generation in Dynamic Networks

Input: Graphs $G = G_1, G_2, \dots, G_T$ such that each snap has same nodes V but different edges in $G_i(V, E_i)$, D = dimension for embedding generation

Output: Quantum Embedding Features

```

1   $NodeEmb_{DW}, NodeEmb_{N2V}, NodeEmb_{WL} = \{\}$   $\triangleright$  each gets empty array of dimension  $|V| * |D * T|$  to store node embeddings of each
   algorithm Deepwalk, Node2Vec and Walklets
2   $NodeEmb_{ALL} = \{\}$   $\triangleright$  empty array of dimension  $|V| * |D * T * 3|$ 
3   $\triangleright$  generating node embeddings for individual random walk-based node embedding algorithms
4  for each snapshot  $t \in 0 \dots T - 1$  do
5    for each algorithm  $ALGO \in \{DW, N2V, WL\}$  do
6       $TempEmb =$  Generate Embeddings using  $ALGO$  for snapshot  $G_{t+1}$ 
7      for each node  $v \in V$  do
8        for each dimension  $d \in 0, \dots, D - 1$  do
9           $NodeEmb_{ALGO}[v][t * D + d] = TempEmb[v][d]$ 
10       end
11     end
12   end
13 end
14  $\triangleright$  concatenating node embeddings for individual random walk-based algorithms into one
15 for each node  $v \in V$  do
16    $NodeEmb_{ALL}[v] = concatenate(NodeEmb_{DW}[v], NodeEmb_{N2V}[v], NodeEmb_{WL}[v])$ 
17 end
18  $E_{TOTAL} \leftarrow E_T \cup$  randomly genrated non-edges of T-th snapshot  $\triangleright$  true and false edges,  $|E_{TOTAL}| = 2 * |E_T|$ 
19  $E_{TRAIN}, E_{TEST} \leftarrow E_{TOTAL}$   $\triangleright$  edge Feature Generation from node embeddings
20  $FeaturesEmbTrain = FeaturesEmbTest = \{\}$   $\triangleright$  dictionaries for train-test edges feature
21 for each edge  $(x, y) \in E_{TOTAL}$  do
22    $TempList = []$   $\triangleright$  temporary empty list to store edge embeddings with * operation
23   for each dimension  $d \in (0, \dots, (D * T * 3) - 1)$  do
24      $TempList = concatenate(TempList, NodeEmb_{ALL}[x][d] * NodeEmb_{ALL}[y][d])$ 
25   end
26   if  $(x, y) \in E_{TRAIN}$  then
27      $FeaturesEmbTrain(x, y) = TempList$ 
28   end
29   else
30      $FeaturesEmbTest(x, y) = TempList$ 
31   end
32 end
33  $FeaturesReduceTrain = PCA(FeaturesEmbTrain)$   $\triangleright$  reducing feature set with PCA
34  $FeaturesReduceTest = PCA(FeaturesEmbTest)$ 
35  $FeaturesTrainPQK, LabelsTrain = PQK(FeaturesReduceTrain)$   $\triangleright$  quantum projection and relabeled features
36  $FeaturesTestPQK, LabelsTest = PQK(FeaturesReduceTest)$ 
37 return  $FeaturesTrainPQK, FeaturesTestPQK, LabelsTrain, LabelsTest$ 

```

algorithm, similarity function, and graph embedding algorithm (Huang, Silva, & Singh, 2021). Quantum walks try to address the first objective efficiently with quantum enhancements. The advantages that we achieve out of quantum walks are: (1) their spreading behaviors are way faster than classical random walks and they span the majority of the vicinity in a shorter time span (Childs et al., 2003) (2) The unitary operator applied for transition is reversible and results in amplitudes instead of probabilities whereas in the classical approach, it is stochastic resulting in random probabilities (Katok & Thouvenot, 2006), (3) quantum walks do not converge to limiting distribution at any point during their evolution (Chakraborty, Luh, & Roland, 2020).

There has been intensive work in this domain, which may be categorized into Discrete-time quantum walk model (DTQWM) (Aharonov, Davidovich, & Zagury, 1993) and Continuous-time quantum walk model (CTQWM) (Farhi & Gutmann, 1998). Based on the concept of coined quantum walk (Wong, 2017), DTQWM is the most widely used model where the evolution takes place in discrete-time steps and is

applied in the discrete scope of positions. The transitions can be defined with the following unitary operator:

$$\hat{U} = \hat{S}(I \otimes \hat{C}) \quad (14)$$

where $\hat{C}$ is the Coin operator, the most commonly used coin operator is the Hadamard operator, and the $\hat{S}$ is the conditional shift operator, the application of the shift operator moves the walker to any direction based on the output of coin operator. This approach has been investigated by many researchers like Nayak and Vishwanath (Ashwin & Ashvin, 2000), Konno (Konno, Namiki, & Soshi, 2004), and Aharonov (Aharonov, Ambainis, Kempe, & Vazirani, 2001).

Continuous-time quantum walk model (CTQWM) is a quantum walk on a graph that is dictated by a time-varying unitary matrix that relies on the Hamiltonian of the quantum system and the adjacency matrix. The term got base-lined when Farhi and Gutmann used it for decision tree traversal (Farhi & Gutmann, 1998). The CTQWM is defined by the unitary operator $\hat{U}$, given by Eq. (15) in which the Hamiltonian is derived from the graphs adjacency matrix.

$$\hat{U} = e^{-iHt} \quad (15)$$

H is the Hamiltonian operator. Now, the position of the walker after time t is denoted by:

$$|\psi(t)\rangle = e^{-iHt} |\psi(0)\rangle \quad (16)$$

Jafarizadeh and Salimi conducted additional research on the dynamics of continuous-time quantum walk evolution on a range of graphs, including cycle, line, complete, and universal graphs (Jafarizadeh & Salimi, 2007). There are numerous other variations of the quantum walk model, such as the Staggered quantum walk model (StQWM) (Portugal, Santos, Fernandes, & Gonçalves, 2016), the Scattering quantum walk model (ScQWM) (Feldman & Hillery, 2004), the Quantum stochastic walk model (QSWM) (Whitfield, Rodríguez-Rosario, & Aspuru-Guzik, 2010), the Open quantum walk model (OQWM) (Attal, Petruccione, Sabot, & Sinayskiy, 2012), and many more. This suggests that much effort has previously been put into improving random walks using quantum enhancements. Our study, however, rather than altering random walks, focuses on improving the embedding generated after the walk via a very innovative projected quantum kernel technique.

4. Proposed work

Quantum machine learning (QML) can refer to three different approaches. The first one is to apply classical machine learning (CML) to solve problems in quantum physics or quantum information. The second is speeding up traditional machine learning techniques using quantum calculations, which occasionally results in hybrid algorithms, also called quantum-enhanced machine learning. The last method is to utilize all of the capabilities of quantum mechanics and quantum information processing by using pure quantum models or adaptations of learning algorithms. Here we employ the second method, where we incorporate quantum enhancement to the random walk-generated features along with a classical machine learning approach for link prediction.

We have introduced a novel approach where we use a projected quantum kernel to project random walk-based embedding vectors into quantum space. This lets us use the quantum benefits of high-dimensional space for better classification. Our work is inspired from the work of *TensorFlow* for quantum feature enhancement (Abadi et al., 2015). The source code for the *TensorFlow* work can be found at open source repository.² The two key components of the algorithm are: (1) Embedding algorithms, as explained previously in Section 3.1 (2) Projected quantum kernel as explained in Sections 2.5 and 2.6. The proposed approach has three phases, as depicted in Fig. 3.

- **Embeddings generation:** The process starts with figuring out features in the form of node embeddings generated by random walk algorithms like Node2Vec, DeepWalk, and Walklet. These are some of the most popular baseline methods. From these node embeddings, edge features are calculated.
- **Quantum Projection with PQK:** Here, quantum enhancements are achieved using the premier model projected quantum kernel (PQK). The Section 2.5 describes how PQK works in its entirety. After embedding generation the features are sent to the PQK, which uses parameterized rotations and entanglement to project them into a quantum high-dimensional Hilbert space where it is easy to separate the data points. Here, they are relabeled and projected back to the classical state. This basically maximizes the geometric distance between two data points and eventually enhances the data classification.
- **Classification and Evaluation:** Finally, the improved features and relabeled data points are fed into a number of classic machine-learning models. The models' performance is judged using a mix of datasets and performance metrics.

The following subsections of the proposed work describe the execution process. The first subsection describes the experimental setup of the embedding algorithms stated earlier. The next one elaborates on the algorithm that we have developed to extract embedding-based features from several snapshots. Subsequently, the execution of the PQK algorithm with the generated features, and finally, the execution of classical ML models with the final projected features.

4.1. Execution of node embedding algorithms used in PQKELP

In this work, we have used three proximity-preserving node embeddings (Rozemberczki, Kiss, & Sarkar, 2020): Node2Vec (Grover & Leskovec, 2016), DeepWalk (Perozzi et al., 2014), and Walklets (Perozzi, Kulkarni, Chen, & Skiena, 2017b).

- **Node2Vec.** Node2Vec is a simple algorithm for learning network node continuous feature representations. It learns such representation in a low-dimensional space that maximizes the likelihood of preserving network neighborhoods. Feature generation for all models is done using the default parameters of Node2Vec. Here we have used $P = 1$ (Return hyperparameter), $Q = 1$ (In-out hyperparameter), $window_size = 10$ (Number of walks per source), $walk_length = 80$ (Length of walk per source), $dimensions = 128$ (Embedding dimension) and $workers = 8$ (No. parallel workers). The feature vector will have a length of $(4 * 128 = 512)$ for five snapshots after embedding the feature representation in 128 dimensions.
- **DeepWalk.** DeepWalk learns latent representations by considering walks as if they were sentences, based on local knowledge gained from shortened random walks. The default DeepWalk parameters are used to generate features for all models. Here we have used $walk_number = 10$, $walk_length = 80$, $dimensions = 128$, $workers = 4$, $window_size = 5$, $epochs = 1$, $learning_rate = 0.05$, $min_count = 1$ and $seed = 4$.
- **Walklets.** Walklets is a technique for learning multi-scale vertices representations that capture multi-scale associations by subsampling short random walks on graph nodes. It generates a corpus of vertex pairings that can be reached through fixed-length pathways by 'skipping' over steps in each random walk. This corpus reveals latent representations encapsulating higher-order relationships in the form of an adjacency matrix. Here we have used default Walklets parameters to generate the feature. The default parameters are $walk_number = 10$, $walk_length = 80$, $dimensions = 32$, $workers = 4$, $window_size = 4$, $epochs = 1$, $learning_rate = 0.05$, $min_count = 1$ and $seed = 42$.

Fig. 2 depicts the structure of the various dynamic networks employed in this paper (left) and their representation of nodes in the embedding space using various Node Embedding techniques such as Node2Vec, DeepWalk, and Walklets (right).

4.2. Proposed quantum embedding algorithm

Algorithm 1 explains the proposed PQKELP (Projected Quantum Kernel Embedding based Link Prediction in Dynamic Networks) framework. In this algorithm, we accept as input the graph. In lines 1–2 we initiate empty arrays for each of the embedding methods namely $DeepWalk(NodeEmb_{DW})$, $Node2Vec(NodeEmb_{N2V})$, and $Walklet(NodeEmb_{WL})$ described in Section 4.1 and an empty array $NodeEmb_{ALL}$ for combined embeddings of all the three. Note that the individual embeddings are of dimension $[V, D * T]$ and the combined embedding is of dimension $[V, D * T * 3]$. In lines 4–13 for every timestamp graph, for each of the random walk algorithms, for every node we append embeddings generated with current algorithms across the row V_i . After the loop, we concatenate the individual embeddings into $NodeEmb_{ALL}$ in lines 12 and 13.

² https://www.tensorflow.org/quantum/tutorials/quantum_data/

The next step is to generate embeddings for each edge of the graph including true edges and randomly generated false edges. In lines 18 and 19 we populated full edge-set E_{TOTAL} with true and false edges and separated the edges into train and test. In lines 20–32, we find edge embeddings by iterating over each edge of E_{TOTAL} and append the product of both the edge-endpoint node embeddings. Then separate these into train and test embeddings based on the edge class namely *FeaturesEmbeddingTrain*, *FeaturesEmbeddingTest*. Because the final dimension tends to be huge i.e. of size $D \times T \times 3$, it is a prerequisite to reduce the dimension in order to make it computable with quantum algorithms. We reduce it with the Principal Component Analysis PCA in lines 27–30. Finally, the last part is to project the features to quantum space, the method POK in lines 33–34, projects the data to quantum space and returns relabeled data back to classical space. Fig. 6 shows the stages of the entire life cycle of the POK algorithm starting from feature projection to dataset relabeling.

4.3. Execution of projected quantum kernel (POK) used in PQKELP

The embedded features made in the previous step are sent to the Projected Quantum Kernel Algorithm, which helps us make the quantum projected data, as we talked about in Section 2.6. The algorithm includes steps for feature projection, data transformation, kernel computation, and relabeling data back to classical with a kernel. This is accomplished with several quantum phases, starting with the generation of a single qubit wall, the creation of a parameterized circuit $V(\hat{\theta})$, the generation of a kernel matrix to get *pqk* features, the computation of eigenvalues and eigenvectors, and the relabeling of datapoints to maximize the geometric distance between kernels. These steps can be visualized in Fig. 6.

In this subsection, we will explain the experimental setup of the projected quantum kernel (POK). 50% node pairs with edges labeled 1 and 50% node pairs with edges labeled 0, make up our data collection. Because quantum computing demands intensive processing, we have conducted experiments on the supercomputer PARAM SHIVAY to accelerate the model's training. Here, normalization is accomplished using the min-max method. Cirq, Sympy, Numpy, Tensorflow, Tensorflow Quantum, and other libraries are among those that are employed to simulate the quantum execution circuits of the POK algorithm. In the data pre-processing step, the dimensions of datasets are reduced to create a data set with fewer dimensions using principal component analysis (PCA). Each feature after PCA is converted into 3D qubits. One extra qubit is taken, so the total qubits are 11. Features after POK = 11×3 (3 for the dimension of qubits) = 33. After this, the reduced features are passed through a quantum circuit for quantum projection, exponentiation, kernel computation, Eigendecomposition, and lastly, relabeling. Hence, as discussed in Section 2.6, a series of quantum algorithms transform the features into a projected quantum kernel.

The classical models have different configurations. The neural network's first layer consists of the feature dimension size input layer. Following that, there was a dense layer of 512 units, a dense layer of 256 units, and then relu activation. The last layer is a dense layer with unit 1 and an activation sigmoid. We use binary cross entropy as a loss function, with metrics being used to measure accuracy and a learning rate of 0.03. The data was trained over 1000 epochs and is supplied in 32-byte batches. For other ML models, default configurations are used. Ten iterations are used to train the entire model, and the average of all iteration results is taken as the final output.

5. Result analysis

This section examines the experimental results received from several combinations of experiments. To begin, we use five performance evaluation metrics: AUC (Liu, Zhang, Lü, & Zhou, 2011b), Precision, Recall, Accuracy (Pedregosa et al., 2011), and F1 score (Goutte & Gaussier, 2005) on five well-known dynamic networks to compare

the proposed framework results in different machine learning algorithms such as Neural Networks (NN), Logistic Regression (LR), XGBoost (XGB), Random Forest Classifier (RFC), and Linear Discriminant Analysis (LDA) with different training-testing ratios. We have studied and compared the results of five training testing ratios (0.5, 0.6, 0.7, 0.8, and 0.9). The notation that is used for a neural network (NN) on an individual quantum random walk embedding with a Projected Quantum Kernel (POK) is QE_{NN} . Similarly, QE_{XGB} , QE_{LR} , QE_{LDA} , QE_{RFC} and QE_{GNB} are used for XGBoost(XGB), Logistic Regression(LR), Linear Discriminant Analysis(LDA), Random Forest Classifier(RFC), and Gaussian's Naive Bayes(GNB). Finally, we mix the features from all the embedding algorithms and compare them to different ML models. The notation followed is $PQKE_{*}$ for each ML model. We also compared the performance of four state-of-the-art methods WEAK (Chiu & Zhan, 2018), Node2Vec (Grover & Leskovec, 2016), CTDNE (Nguyen et al., 2018), and LGQ (Kumar et al., 2021), to propose a link prediction framework with different machine learning variations. The test results showed that the suggested framework for predicting links works better on five well-known dynamic networks with five assessment metrics when mixed with different machine learning algorithms. The following sections and tables show how well POK-based link prediction works when node embedding and different machine-learning models are used together.

5.1. Datasets

We have evaluated our results on five well-known datasets. The datasets are MIT³ (Kunegis, 2017) (human interaction data of 100 MIT students), Radoslaw-Email⁴ (Michalski, Palus, & Kazienko, 2011) (email transmission between company employees), EU-Core⁵ (email transmission with an European institution), FB-Forum⁶ (Rossi & Ahmed, 2015) (open interactions in a social network), and, CollegeMsg⁷ (student engagement on a social media platform). These temporal networks, available on Leskovec and Krevl (2014), with varying graph sizes, densities, and time periods. Alongside, various network data statistics⁸ are shown in Table 1. Also, Table 2 shows mean, variance and average clustering coefficient of dataset at different snapshot. This will help us understand the impact on the performance of varying network sizes, densities, clustering, etc.

5.2. Performance comparison of proposed quantum embedding with neural network (QE_{NN}) on different random-walk techniques and different train-test ratio

This section start with a Performance Comparison of Link Prediction in a Neural Network (NN) with a Projected Quantum Kernel (QE_{NN}) on five well-known and publicly available datasets with three different embedding algorithm-based features and considering different training and testing ratios. The performance of all these approaches is compared using four assessment metrics: AUC, Precision, Recall, Accuracy, and F1 score. Table 3 shows the performance comparison of Neural Network (NN) based Embedded Projected Quantum Kernel (QE_{NN}) with different embedding techniques and considering different training and testing ratios. The AUC metric where in the case of the mit dataset Deepwalk embedding method outperforms other methods with the highest score on a ratio of 0.5. On radoslaw-email Deepwalk and Walklets also perform competitively with the highest scores on ratios of 0.7 and 0.9 respectively. For Eu-Core and fb-forum, Node2Vec is better

³ <http://konect.cc/networks/mit/>

⁴ <http://networkrepository.com/ia-radoslaw-email.php>

⁵ <https://snap.stanford.edu/data/email-Eu-core.html>

⁶ <http://networkrepository.com/fb-forum.php>

⁷ <https://snap.stanford.edu/data/CollegeMsg.html>

⁸ <https://networkrepository.com/>

Table 1
Dataset statistics.

Datasets	V	E	Avg. degree	dmax	Clustering coefficient	Number of triangles
mit	96	1,086,404	25	92	0.725376	36108
fb-forum	899	33686	7	128	0.155982	4659
radoslaw-email	167	82876	22	139	4.50961	37209
Eu-core	986	332334	16	345	0.3994	105461
CollegeMsg	1899	59835	6	255	0.0568303	14319

Table 2
Dataset Statistics at different Snapshot (Degree mean, variance and clustering).

Dataset	Metric	Snapshot 1	Snapshot 2	Snapshot 3	Snapshot 4	Snapshot 5	Overall
mit	Mean Degree	38	39	19	16	12	25
	Degree Variance	314	279	208	170	122	219
	Avg. Clustering coef.	0.7524	0.7873	0.7526	0.7085	0.6847	0.7171
fb-forum	Mean Degree	14	6	5	4	3	7
	Degree Variance	229	33	43	25	18	70
	Avg. Clustering coef.	0.1013	0.1158	0.1593	0.0905	0.1097	0.1273
radoslaw-email	Mean Degree	27	25	18	18	20	22
	Degree Variance	566	478	196	179	250	334
	Avg. Clustering coef.	4.5908	4.5345	4.3596	4.8166	4.4463	4.5626
Eu-core	Mean Degree	20	20	21	14	4	16
	Degree Variance	440	514	534	231	29	350
	Avg. Clustering coef.	0.3562	0.3682	0.3592	0.3458	0.2047	0.3268
CollegeMsg	Mean Degree	12	8	4	3	3	6
	Degree Variance	402	118	54	33	16	125
	Avg. Clustering coef.	0.0189	0.0740	0.0537	0.0728	0.0671	0.0653

Table 3
Performance comparison of proposed quantum random walk embedding with neural network (QE_NN) on different random walk techniques and different train-test ratio.

Metric	Datasets	Node2Vec	DeepWalk	Walklets	Node2Vec	DeepWalk	Walklets	Node2Vec	DeepWalk	Walklets	Node2Vec	DeepWalk	Walklets	Node2Vec	DeepWalk	Walklets
		0.5			0.6			0.7			0.8			0.9		
AUC	mit	0.5792	0.8796	0.8572	0.5729	0.8570	0.8274	0.6010	0.8760	0.8514	0.6007	0.8696	0.8582	0.5551	0.8461	0.8072
	radoslaw-email	0.4896	0.9218	0.9033	0.5651	0.9211	0.9160	0.5803	0.9244	0.9063	0.5774	0.9132	0.9089	0.6402	0.9150	0.9310
	Eu-core	0.9389	0.9125	0.8296	0.9419	0.9026	0.7500	0.9431	0.7768	0.8395	0.9210	0.9034	0.8385	0.9354	0.8842	0.9108
	fb-forum	0.9299	0.8898	0.8462	0.9321	0.8957	0.8555	0.9294	0.9054	0.8772	0.9540	0.8891	0.8394	0.9315	0.8836	0.8720
	CollegeMsg	0.6697	0.7971	0.9465	0.6683	0.8243	0.9480	0.6673	0.7847	0.9454	0.5134	0.8267	0.9529	0.6680	0.9121	0.9505
Precision	mit	0.3943	0.7876	0.7945	0.2183	0.8258	0.7732	0.2038	0.7973	0.8235	0.4286	0.7563	0.7735	0.3104	0.7545	0.7763
	radoslaw-email	0.0000	0.8553	0.8021	0.1912	0.8803	0.8434	0.1865	0.8133	0.8839	0.1742	0.8767	0.8170	0.3453	0.8613	0.9052
	Eu-core	0.8529	0.8904	0.6599	0.8673	0.7323	0.5758	0.8888	0.5595	0.8111	0.8826	0.7885	0.7435	0.8933	0.8343	0.8604
	fb-forum	0.8569	0.8232	0.7958	0.8851	0.8322	0.7856	0.8810	0.8532	0.8029	0.8544	0.8215	0.8162	0.8972	0.8154	0.7730
	CollegeMsg	0.4191	0.7738	0.9210	0.4532	0.6698	0.9224	0.3379	0.8528	0.8872	0.1808	0.8120	0.9075	0.1690	0.8284	0.9354
Recall	mit	0.8000	0.8384	0.7744	0.3960	0.7168	0.7262	0.4000	0.8395	0.7075	0.7552	0.7778	0.7014	0.5842	0.9348	0.7973
	radoslaw-email	0.0000	0.8379	0.8657	0.0847	0.8291	0.8460	0.1156	0.9016	0.7406	0.1696	0.7957	0.8609	0.3043	0.8530	0.8276
	Eu-core	0.8918	0.7943	0.8298	0.9115	0.8817	0.5448	0.9234	0.5243	0.7354	0.8392	0.9014	0.7301	0.8947	0.8461	0.8142
	fb-forum	0.9033	0.8085	0.7672	0.8634	0.8075	0.7794	0.8557	0.8155	0.7919	0.9233	0.8219	0.7010	0.8292	0.8351	0.8236
	CollegeMsg	0.5180	0.6333	0.8696	0.4744	0.8407	0.8910	0.5396	0.5555	0.9015	0.1681	0.5139	0.9096	0.1885	0.8773	0.8755
Accuracy	mit	0.4897	0.8032	0.7854	0.5073	0.7759	0.7602	0.5919	0.8100	0.7670	0.5316	0.7633	0.7815	0.5072	0.7879	0.7612
	radoslaw-email	0.9528	0.8434	0.8208	0.8992	0.8554	0.8444	0.9153	0.8433	0.8172	0.9322	0.8389	0.8370	0.8967	0.8487	0.8612
	Eu-core	0.8645	0.8447	0.7236	0.8831	0.7796	0.8558	0.9021	0.8514	0.7636	0.8600	0.8333	0.7526	0.8940	0.8272	0.8318
	fb-forum	0.8728	0.8199	0.7838	0.8743	0.8208	0.7851	0.8691	0.8368	0.7985	0.8820	0.8100	0.7543	0.8708	0.8194	0.8090
	CollegeMsg	0.8266	0.6781	0.8983	0.8168	0.7024	0.9032	0.7911	0.6700	0.8965	0.8203	0.6675	0.9010	0.9408	0.8520	0.9082
F1 score	mit	0.5282	0.8122	0.7843	0.2815	0.7675	0.7489	0.2700	0.8179	0.7611	0.5468	0.7669	0.7356	0.4054	0.8351	0.7867
	radoslaw-email	0.0000	0.8465	0.8327	0.1174	0.8539	0.8447	0.1427	0.8552	0.8059	0.1719	0.8342	0.8384	0.3235	0.8572	0.8647
	Eu-core	0.8720	0.8396	0.7352	0.8888	0.8000	0.5599	0.9058	0.5413	0.7714	0.8603	0.8412	0.7367	0.8940	0.8402	0.8366
	fb-forum	0.8795	0.8157	0.7812	0.8741	0.8196	0.7825	0.8681	0.8339	0.7974	0.8875	0.8217	0.7542	0.8619	0.8251	0.7975
	CollegeMsg	0.4633	0.6965	0.8946	0.4635	0.7456	0.9064	0.4155	0.6728	0.8943	0.1743	0.6294	0.9086	0.1782	0.8522	0.9045

with the highest on 0.6 and 0.8 respectively. Walklet is best for CollegeMsg with the highest score on 0.9. In the case of Precision, for mit and radoslaw-email Deepwalk and Walklets perform competitively with the individual highest on 0.6 & 0.7 and 0.6 & 0.9 ratio respectively. For Eu-core and fb-forum Node2Vec outperforms with the highest score on 0.9. For CollegeMsg Walklets give better results with the highest on ratio of 0.9. Considering recall for mit Deepwalk is better with the highest on 0.9. For radoslaw-email Deepwalk and Walklets perform competitively with the highest scores on 0.7 & 0.5 respectively. For eu-core and fb-forum again N2V is best with highest scores on ratio of 0.7 and 0.8 respectively. For CollegeMsg again Walklets outperform with highest on 0.9 ratio. For accuracy, N2V outperforms on radoslaw-email, fb-forum, and Eu-core with the highest scores on 0.5, 0.7, and 0.8 respectively. For mit Deepwalk and for CollegeMsg Walklets are better with highest on ratios 0.5 and 0.9 respectively. Lastly in the case of F1 Score Deepwalk gives better results for mit and radoslaw-email having highest score on 0.9 ratio. N2V gives better results for Eu-core and fb-forum having highest scores on 0.7 and 0.8 respectively. Lastly, Walklets are best for CollegeMsg on 0.8.

5.3. Performance comparison of proposed quantum embedding with logistic regression (QE_LR) on different random-walk techniques and different train-test ratio

This section compares the performance of Link Prediction in Logistic Regression (LR) with Embedded Projected Quantum Kernel (QE_LR) on various training and testing ratios with different embedding methods. The analysis and comparison is depicted in Table 4.

In terms of AUC Deepwalk embeddings outperform other methods for mit and radoslaw-email with the highest scores on 0.9 ratio for both networks. Node2Vec is better on Eu-core and fb-forum having the highest at 0.7 and 0.8 ratio respectively. Walklets perform best on CollegeMsg with the highest on ratio of 0.9. Precision is high on mit and radoslaw-email for Deepwalk on 0.9 ratio. For Eu-core and fb-forum Node2Vec is the winner with the highest on 0.7. Walklet is the highest amongst other embedding methods for CollegeMsg at 0.9 ratio. Considering recall Deepwalk is again highest for mit and radoslaw-email with top scores on 0.9 and 0.6, Node2Vec for Eu-core, fb-forum

Table 4

Performance comparison of proposed quantum random walk embedding with linear regression (*QE_LR*) on different random walk techniques and different train-test ratio.

Metric	Datasets	Node2Vec	DeepWalk	Walklets	Node2Vec	DeepWalk	Walklets	Node2Vec	DeepWalk	Walklets	Node2Vec	DeepWalk	Walklets	Node2Vec	DeepWalk	Walklets
		0.5			0.6			0.7			0.8			0.9		
AUC	mit	0.5593	0.6908	0.6699	0.5675	0.6624	0.6455	0.6025	0.6126	0.6468	0.5560	0.6817	0.6523	0.6056	0.7170	0.6284
	radoslaw-email	0.4979	0.7075	0.6787	0.5632	0.7178	0.6735	0.5319	0.7198	0.6878	0.5780	0.6938	0.6837	0.6130	0.7256	0.7018
	Eu-core	0.9198	0.7498	0.5526	0.9251	0.6506	0.6523	0.9355	0.6344	0.6194	0.9007	0.7678	0.6013	0.9255	0.6651	0.6928
	fb-forum	0.9146	0.7422	0.5963	0.9199	0.7585	0.6077	0.9117	0.7481	0.6330	0.9338	0.6968	0.6130	0.9198	0.7973	0.6162
	CollegeMsg	0.6624	0.5905	0.9174	0.6711	0.6527	0.9159	0.6716	0.6293	0.9122	0.5267	0.7721	0.9104	0.6046	0.7431	0.9132
Precision	mit	0.5526	0.6482	0.6043	0.5804	0.6350	0.5964	0.4912	0.5939	0.6252	0.5361	0.6073	0.5456	0.6282	0.7202	0.6324
	radoslaw-email	0.0446	0.6589	0.6437	0.2091	0.6654	0.6234	0.1860	0.6816	0.6514	0.1994	0.6478	0.6085	0.3286	0.6935	0.6730
	Eu-core	0.8591	0.7261	0.4929	0.8575	0.6090	0.4899	0.8791	0.4587	0.6289	0.8548	0.6953	0.5161	0.8658	0.6584	0.6353
	fb-forum	0.8674	0.6780	0.5803	0.8511	0.6858	0.5495	0.8538	0.6957	0.5575	0.8434	0.6687	0.6137	0.8358	0.7424	0.5419
	CollegeMsg	0.4387	0.6483	0.8580	0.4722	0.5708	0.8700	0.3656	0.6684	0.8280	0.0588	0.6398	0.8503	0.2202	0.7018	0.8365
Recall	mit	0.5561	0.6532	0.6044	0.5550	0.6165	0.6113	0.6492	0.5636	0.6065	0.6431	0.6125	0.6079	0.6224	0.6742	0.5811
	radoslaw-email	0.4834	0.6392	0.6279	0.5029	0.6856	0.6352	0.5601	0.6407	0.6389	0.5810	0.6094	0.6626	0.6104	0.6625	0.6497
	Eu-core	0.8602	0.7171	0.5913	0.8757	0.6218	0.5580	0.8856	0.5990	0.5837	0.8399	0.7465	0.6192	0.8821	0.6272	0.5796
	fb-forum	0.8338	0.6826	0.5760	0.8503	0.7046	0.5731	0.8403	0.6859	0.5375	0.8819	0.6642	0.6043	0.8471	0.6682	0.6102
	CollegeMsg	0.8187	0.4625	0.8594	0.7835	0.6751	0.8357	0.8936	0.6185	0.8727	0.2949	0.5569	0.8274	0.4780	0.6868	0.8981
Accuracy	mit	0.5544	0.6439	0.6020	0.5609	0.6231	0.6085	0.5853	0.5813	0.6098	0.5689	0.6127	0.6000	0.6123	0.6591	0.5821
	radoslaw-email	0.4879	0.6499	0.6379	0.5883	0.6654	0.6264	0.5745	0.6651	0.6401	0.5589	0.6346	0.6248	0.6145	0.6648	0.6417
	Eu-core	0.8608	0.7122	0.5347	0.8638	0.6285	0.6756	0.8792	0.5975	0.5921	0.8475	0.7126	0.5569	0.8710	0.6313	0.6129
	fb-forum	0.8516	0.6835	0.5778	0.8502	0.6910	0.5569	0.8485	0.6912	0.5579	0.8620	0.6476	0.5865	0.8417	0.7118	0.5799
	CollegeMsg	0.5573	0.5623	0.8620	0.5410	0.6062	0.8493	0.5350	0.5878	0.8526	0.6863	0.5939	0.8274	0.6939	0.7143	0.8622
F1 score	mit	0.5543	0.6507	0.6043	0.5674	0.6256	0.6038	0.5592	0.5784	0.6157	0.5848	0.6099	0.5751	0.6253	0.6964	0.6056
	radoslaw-email	0.0816	0.6489	0.6357	0.2954	0.6754	0.6293	0.2792	0.6605	0.6451	0.2969	0.6280	0.6344	0.4273	0.6776	0.6611
	Eu-core	0.8596	0.7216	0.5377	0.8665	0.6153	0.5217	0.8823	0.5195	0.6055	0.8473	0.7200	0.5630	0.8739	0.6424	0.6062
	fb-forum	0.8503	0.6803	0.5782	0.8507	0.6951	0.5611	0.8470	0.6908	0.5473	0.8622	0.6664	0.6090	0.8414	0.7033	0.5740
	CollegeMsg	0.5713	0.5398	0.8587	0.5892	0.6186	0.8525	0.5190	0.6425	0.8497	0.0981	0.5955	0.8387	0.3015	0.6942	0.8662

at 0.7 and 0.8, and Walklets for CollegeMsg on 0.9 ratio. Performance of accuracy and F1 score is very similar to AUC with the same models and ratios except Walklet being highest on 0.9 for CollegeMsg.

5.4. Performance comparison of proposed quantum embedding with linear discriminant analysis (*QE_LDA*) on different random-walk techniques and different train-test ratio

This section compares the performance of Link Prediction in Linear Discriminant Analysis (*LDA*) machine learning classifier and Embedded Projected Quantum Kernel (*QE_LDA*) on various training and testing ratios with different embedding methods as shown in Table 5

In terms of AUC, Node2Vec embedding is prominent over other methods for 3 networks. The highest scores for N2V are on the ratio of 0.8 for mit, 0.7 for Eu-core, and 0.6 for fb-forum. For radoslaw-email Deepwalk besides having a better score N2V steals the highest score on ratio of 0.9. For CollegeMsg, Walklet as usual is best at 0.9 ratio. Precision is high for Node2Vec on mit at 0.9, Eu-core at 0.7, and fb-forum at 0.5. For radoslaw-email, Deepwalk outperforms with highest score on ratio of 0.9. Walklets high on CollegeMsg at 0.6. Considering recall Node2Vec is again highest for mit and Eu-core and fb-forum with highest score on ratio of 0.8, 0.7 and 0.8 respectively. Deepwalk and Walklets perform competitively on radoslaw-email having Deepwalk highest on 0.6. CollegeMsg has Walklet performing better with highest on 0.9 ratio. For Accuracy and F1 score the score patterns are similar where Node2Vec outperforms with highest score on ratio of 0.6 for mit, 0.9 for radoslaw-email, 0.7 for Eu-core, and 0.8 for fb-forum and lastly, Walklet outperforming on ratio of 0.9.

5.5. Performance comparison of proposed quantum embedding with XGBoost (*QE_XGB*) on different random-walk techniques and different train-test ratio

On various training and testing ratios, this section analyzes the performance of Link Prediction in the XGBoost (*XGB*) machine learning classifier and Embedded Projected Quantum Kernel (*QE_XGB*). These algorithms are evaluated using four assessment metrics and based on five well-known and publicly available datasets on a variety of training and testing ratios with different embedding methods, as depicted in Table 6

In terms of AUC, Node2Vec embedding is prominent over other methods for three networks. The highest scores are on a ratio of 0.8 for mit, 0.9 for Eu-core and fb-forum. For radoslaw-email, Deepwalk does better and takes the highest score on a ratio of 0.7. For CollegeMsg, Walklet being the consistent performer has the highest score at 0.9 ratio

for all metrics except Precision on 0.8 ratio. In terms of Precision on other networks, it is high for Node2Vec on mit, Eu-core, and fb-forum all on 0.9 ratio. On radoslaw-email, Deepwalk and Walklet perform competitively with Walklet having the highest on 0.9 and Deepwalk on 0.7 ratio. Considering recall Node2Vec is again highest for mit and Eu-core and fb-forum with highest scores on 0.7, 0.9, and 0.8 respectively. Deepwalk outperforms for radoslaw-email having highest on 0.7. For Accuracy and F1 score the score patterns are similar, here Node2Vec outperforms with the highest score on ratio of 0.8 for mit, 0.9 for Eu-core, and 0.9 for fb-forum, whereas Deepwalk is better for radoslaw email having highest on ratio of 0.9 for accuracy and 0.7 for F1 score.

5.6. Performance comparison of proposed quantum embedding with random forest classifier (*QE_RFC*) on different random-walk techniques and different train-test ratio

On various training and testing ratios, this section analyzes the performance of Link Prediction in the Random Forest Classifier (*RFC*) and Embedded Projected Quantum Kernel (*QE_RFC*). These algorithms are evaluated using four assessment metrics and based on five well-known and publicly available datasets On a variety of training and testing ratios with different embedding methods, as depicted in Table 7

In terms of AUC, Node2Vec embedding is prominent over other methods for three networks. The highest scores are on ratio of 0.8 for mit, 0.9 for Eu-core, and 0.8 for fb-forum. For radoslaw-email, Deepwalk does better and takes the highest score on ratio of 0.8 ratio. For CollegeMsg, Walklet being the consistent performer on the CollegeMsg network has the highest score at different ratios for different metrics. In terms of Precision, it is high for Deepwalk on radoslaw-email on 0.6 ratio, Eu-core, and fb-forum on 0.9 ratio. On radoslaw-email, Deepwalk performs best having highest on 0.6 ratio. Considering recall Node2Vec is again highest for mit and Eu-core and fb-forum with highest score on ratio of 0.8, 0.9 and 0.7 respectively. Deepwalk outperforms for radoslaw-email having highest on 0.9. For Accuracy and F1 score the score patterns are identical, here Node2Vec outperforms with highest score on ratio of 0.8 for mit, 0.9 for Eu-core and fb-forum, whereas Deepwalk and Walklets perform competitively for radoslaw-email having highest on 0.9.

5.7. Performance comparison of proposed quantum embedding with Gaussian's Naive Bayes (*QE_GNB*) on different random-walk techniques and different train-test ratio

On various training and testing ratios, this section analyzes the performance of Link Prediction in the Gaussian Naive Bayes (*GNB*)

Table 5

Performance comparison of proposed quantum random walk embedding with linear discriminant analysis (*QE_LDA*) on different random walk techniques and different train–test ratio.

Metric	Datasets	Node2Vec	DeepWalk	Walklets	Node2Vec	DeepWalk	Walklets	Node2Vec	DeepWalk	Walklets	Node2Vec	DeepWalk	Walklets	Node2Vec	DeepWalk	Walklets
		0.5			0.6			0.7			0.8			0.9		
AUC	mit	0.8991	0.6873	0.6776	0.8852	0.6607	0.6403	0.7472	0.6183	0.6393	0.9049	0.6814	0.6501	0.8934	0.7242	0.6144
	radoslaw-email	0.4814	0.7076	0.6817	0.5923	0.7178	0.6725	0.5848	0.7199	0.6892	0.5870	0.6942	0.6841	0.7568	0.7261	0.7019
	Eu-core	0.9143	0.7556	0.5505	0.9207	0.6507	0.6500	0.9296	0.6421	0.6176	0.9004	0.7707	0.5971	0.9203	0.6675	0.6924
	fb-forum	0.9147	0.7449	0.5973	0.9176	0.7567	0.6040	0.9078	0.7430	0.6330	0.9303	0.6987	0.6160	0.9155	0.7993	0.6082
	CollegeMsg	0.6510	0.5980	0.9149	0.6704	0.6515	0.9153	0.6911	0.6253	0.9075	0.5257	0.7682	0.9101	0.5986	0.7386	0.9115
Precision	mit	0.8373	0.6517	0.6304	0.8405	0.6292	0.5861	0.6072	0.5990	0.6165	0.8006	0.6096	0.5518	0.8557	0.7296	0.6310
	radoslaw-email	0.0458	0.6636	0.6442	0.2223	0.6615	0.6234	0.1996	0.6792	0.6526	0.2089	0.6505	0.6097	0.3869	0.6956	0.6755
	Eu-core	0.8487	0.7212	0.4902	0.8481	0.5998	0.4885	0.8701	0.4647	0.6223	0.8537	0.6962	0.5173	0.8409	0.6504	0.6300
	fb-forum	0.8653	0.6811	0.5778	0.8544	0.6900	0.5536	0.8540	0.6941	0.5508	0.8411	0.6666	0.6282	0.8193	0.7499	0.5338
	CollegeMsg	0.4289	0.6451	0.8527	0.4685	0.5766	0.8724	0.3682	0.6743	0.8117	0.0684	0.6489	0.8591	0.2070	0.7083	0.8535
Recall	mit	0.8022	0.6532	0.6344	0.8201	0.6137	0.6038	0.6644	0.5739	0.5774	0.8519	0.5958	0.5763	0.7795	0.6742	0.5541
	radoslaw-email	0.3689	0.6433	0.6268	0.4551	0.6812	0.6321	0.4731	0.6424	0.6407	0.4136	0.6118	0.6688	0.6542	0.6688	0.6425
	Eu-core	0.8476	0.7089	0.5881	0.8646	0.6238	0.5447	0.8712	0.6148	0.5789	0.8341	0.7347	0.6219	0.8713	0.6142	0.5752
	fb-forum	0.8349	0.6847	0.5696	0.8366	0.6957	0.5787	0.8198	0.6813	0.5361	0.8728	0.6579	0.6204	0.8505	0.6961	0.5896
	CollegeMsg	0.5478	0.4586	0.8539	0.6479	0.6759	0.8196	0.6188	0.6064	0.8725	0.2863	0.5489	0.8315	0.2780	0.7082	0.8981
Accuracy	mit	0.8217	0.6469	0.6286	0.8224	0.6181	0.5986	0.7331	0.5862	0.5966	0.8190	0.6128	0.6000	0.8192	0.6667	0.5746
	radoslaw-email	0.6129	0.6542	0.6381	0.6764	0.6612	0.6257	0.6640	0.6638	0.6414	0.6879	0.6365	0.6268	0.7128	0.6686	0.6417
	Eu-core	0.8489	0.7073	0.5317	0.8528	0.6224	0.6898	0.8667	0.6026	0.5859	0.8443	0.7103	0.5581	0.8507	0.6198	0.6083
	fb-forum	0.8505	0.6863	0.5746	0.8458	0.6916	0.5609	0.8393	0.6889	0.5517	0.8564	0.6442	0.6021	0.8319	0.7257	0.5694
	CollegeMsg	0.7801	0.5596	0.8566	0.6395	0.6096	0.8434	0.7303	0.5890	0.8425	0.7188	0.5964	0.8350	0.8224	0.7245	0.8724
F1 score	mit	0.8194	0.6525	0.6324	0.8302	0.6213	0.5948	0.6345	0.5862	0.5963	0.8254	0.6026	0.5638	0.8158	0.7008	0.5900
	radoslaw-email	0.0815	0.6533	0.6354	0.2987	0.6712	0.6277	0.2808	0.6603	0.6466	0.2776	0.6306	0.6379	0.4862	0.6819	0.6586
	Eu-core	0.8481	0.7150	0.5347	0.8562	0.6116	0.5151	0.8706	0.5293	0.5998	0.8438	0.7150	0.5648	0.8558	0.6318	0.6014
	fb-forum	0.8498	0.6829	0.5737	0.8454	0.6928	0.5659	0.8366	0.6876	0.5433	0.8567	0.6622	0.6243	0.8346	0.7220	0.5603
	CollegeMsg	0.4811	0.5361	0.8533	0.5438	0.6223	0.8451	0.4617	0.6386	0.8410	0.1104	0.5947	0.8450	0.2373	0.7083	0.8752

Table 6

Performance comparison of proposed quantum random walk embedding with XGBoost (*QE_XGB*) on different random walk techniques and different train–test ratio.

Metric	Datasets	Node2Vec	DeepWalk	Walklets	Node2Vec	DeepWalk	Walklets	Node2Vec	DeepWalk	Walklets	Node2Vec	DeepWalk	Walklets	Node2Vec	DeepWalk	Walklets
		0.5			0.6			0.7			0.8			0.9		
AUC	mit	0.8300	0.7150	0.7345	0.8578	0.6800	0.6781	0.8109	0.6734	0.7274	0.8996	0.7067	0.7293	0.8731	0.7583	0.6613
	radoslaw-email	0.4997	0.7441	0.7479	0.5786	0.7384	0.7454	0.5761	0.7568	0.7450	0.5799	0.7444	0.7555	0.6594	0.7222	0.7724
	Eu-core	0.8383	0.7630	0.6644	0.8382	0.6923	0.6864	0.8302	0.6577	0.7325	0.8130	0.7614	0.7242	0.8607	0.7655	0.7877
	fb-forum	0.8235	0.7508	0.7418	0.8267	0.7579	0.7505	0.8264	0.7586	0.7839	0.8383	0.7593	0.7608	0.8462	0.7869	0.7318
	CollegeMsg	0.6616	0.6118	0.8475	0.6679	0.6536	0.8688	0.6481	0.6146	0.8634	0.5035	0.6855	0.8525	0.5736	0.7529	0.8724
Precision	mit	0.8580	0.7170	0.7581	0.8835	0.6843	0.6516	0.7311	0.7084	0.7503	0.8966	0.7050	0.7051	0.8850	0.8248	0.7077
	radoslaw-email	0.0295	0.7666	0.7729	0.4232	0.7490	0.7517	0.1796	0.7748	0.7713	0.2063	0.7644	0.7819	0.3867	0.7576	0.7760
	Eu-core	0.8385	0.7665	0.6032	0.8457	0.6793	0.5336	0.8334	0.5248	0.7681	0.8206	0.7464	0.6877	0.8444	0.8315	0.8011
	fb-forum	0.8306	0.7538	0.7539	0.8355	0.7352	0.7515	0.8311	0.7819	0.8008	0.8255	0.7604	0.8634	0.8501	0.8238	0.7460
	CollegeMsg	0.4401	0.7153	0.8260	0.4696	0.6257	0.8646	0.3580	0.7337	0.8640	0.0517	0.7191	0.8814	0.2016	0.7655	0.8523
Recall	mit	0.7892	0.7253	0.7008	0.8370	0.6970	0.7304	0.9028	0.6218	0.7157	0.8974	0.6931	0.6913	0.8640	0.7373	0.6892
	radoslaw-email	0.6000	0.7254	0.7064	0.3748	0.7380	0.7385	0.1762	0.7465	0.7170	0.7840	0.7431	0.6981	0.9612	0.7060	0.8318
	Eu-core	0.8334	0.7947	0.7567	0.8299	0.6809	0.5094	0.8341	0.8143	0.7199	0.8041	0.7840	0.7447	0.8842	0.7293	0.7965
	fb-forum	0.8177	0.7378	0.7316	0.8123	0.8080	0.7518	0.8179	0.7328	0.7644	0.8482	0.8046	0.6366	0.8397	0.7383	0.6607
	CollegeMsg	0.9110	0.4534	0.8695	0.8720	0.6976	0.8887	0.9310	0.5130	0.8526	0.7385	0.5699	0.8368	0.7885	0.7268	0.8962
Accuracy	mit	0.8299	0.7151	0.7351	0.8568	0.6822	0.6776	0.7230	0.6747	0.7256	0.8995	0.7064	0.7296	0.8718	0.7576	0.6642
	radoslaw-email	0.4112	0.7436	0.7474	0.7559	0.7384	0.7460	0.9378	0.7574	0.7454	0.4010	0.7483	0.7560	0.3893	0.7241	0.7803
	Eu-core	0.8383	0.7648	0.6575	0.8383	0.6878	0.8374	0.8303	0.4993	0.7319	0.8136	0.7609	0.7192	0.8599	0.7719	0.7880
	fb-forum	0.8236	0.7520	0.7421	0.8265	0.7582	0.7500	0.8264	0.7586	0.7854	0.8378	0.7616	0.7526	0.8458	0.7847	0.7361
	CollegeMsg	0.4655	0.5946	0.8465	0.4547	0.6425	0.8695	0.4323	0.5946	0.8639	0.2964	0.6472	0.8503	0.4041	0.7602	0.8724
F1 score	mit	0.8222	0.7211	0.7284	0.8596	0.6906	0.6887	0.8080	0.6623	0.7326	0.8970	0.6990	0.6981	0.8744	0.7786	0.6983
	radoslaw-email	0.0563	0.7454	0.7381	0.3976	0.7434	0.7450	0.1779	0.7604	0.7431	0.3266	0.7536	0.7376	0.5515	0.7309	0.8029
	Eu-core	0.8360	0.7803	0.6713	0.8377	0.6801	0.5212	0.8337	0.6382	0.7432	0.8122	0.7647	0.7151	0.8638	0.7770	0.7988
	fb-forum	0.8241	0.7457	0.7426	0.8238	0.7699	0.7517	0.8244	0.7566	0.7822	0.8367	0.7819	0.7328	0.8449	0.7787	0.7008
	CollegeMsg	0.5935	0.5550	0.8472	0.6104	0.6597	0.8765	0.5172	0.6038	0.8583	0.0967	0.6359	0.8585	0.3211	0.7456	0.8737

Table 7

Performance comparison of proposed quantum random walk embedding with random forest classifier (*QE_RFC*) on different random walk techniques and different train–test ratio.

Metric	Datasets	Node2Vec	DeepWalk	Walklets	Node2Vec	DeepWalk	Walklets	Node2Vec	DeepWalk	Walklets	Node2Vec	DeepWalk	Walklets	Node2Vec	DeepWalk	Walklets
		0.5			0.6			0.7			0.8			0.9		
AUC	mit	0.9198	0.8701	0.8439	0.9295	0.8327	0.8308	0.8361	0.8401	0.8409	0.9570	0.8578	0.8247	0.9283	0.8174	0.7851
	radoslaw-email	0.4851	0.9035	0.8939	0.5734	0.9039	0.9062	0.5612	0.9017	0.8989	0.5572	0.9076	0.8960	0.6760	0.8906	0.9256
	Eu-core	0.9322	0.9091	0.7619	0.9318	0.8271	0.7444	0.9320	0.7459	0.7986	0.9100	0.9169	0.8056	0.9325	0.9016	0.9129
	fb-forum	0.9219	0.8911	0.8837	0.9263	0.9011	0.8853	0.9220	0.9112	0.8999	0.9408	0.8842	0.8852	0.9305	0.9293	0.8988
	CollegeMsg	0.6776	0.7257	0.9319	0.6651	0.7487	0.9403	0.6457	0.7312	0.9376	0.5362	0.8831	0.9408	0.5513	0.9148	0.9335
Precision	mit	0.8735	0.8033	0.7808	0.9076	0.7805	0.7605	0.7391	0.7832	0.7840	0.9121	0.7309	0.6893	0.8907	0.8028	0.8074
	radoslaw-email	0.0456	0.8482	0.8210	0.2216	0.8623	0.8223	0.2196	0.8205	0.8303	0.2174	0.8357	0.8201	0.3896	0.8225	0.8744
	Eu-core	0.8692	0.8423	0.6418	0.9004	0.7250	0.5836	0.8839	0.5792	0.8329	0.8625	0.8137	0.7435	0.8870	0.8682	0.8435
	fb-forum	0.8663	0.8017	0.8154	0.8775	0.8803	0.8073	0.8523	0.8343	0.8409	0.8786	0.8511	0.8557	0.8843	0.8765	0.8011
	CollegeMsg	0.4424	0.7618	0.8274	0.4725	0.6575	0.8688	0.3616	0.7677	0.9119	0.0809	0.8186	0.9449	0.1844	0.8091	0.8613
Recall	mit	0.8618	0.8047	0.7858	0.8709	0.7293	0.7684	0.7702	0.7891	0.7785	0.8938	0.8278	0.7880	0.8868	0.8066	0.6622
	radoslaw-email	0.3398	0.8068	0.8219	0.4492	0.7947	0.8361	0.4617	0.8390	0.8163	0.4037	0.8286	0.8576	0.5638	0.7758	0.8648
	Eu-core	0.8812	0.8428	0.7557	0.8536	0.8289	0.6507	0.8700	0.6033	0.7669	0.8517	0.8803	0.7817	0.8865	0.8431	0.8496
	fb-forum	0.8747	0.8183	0.7834	0.8720	0.7991	0.7996	0.8847	0.8443	0.8116	0.8694	0.7785	0.7307	0.8572	0.8714	0.8125
	CollegeMsg	0.8560	0.5649	0.9297	0.7795	0.7766	0.9199	0.8136	0.7156	0.8478	0.3294	0.9420	0.8037	0.2170	0.8491	0.8987
Accuracy	mit	0.8677	0.8002	0.7804	0.8860	0.7549	0.7687	0.8341	0.7778	0.7753	0.9066	0.7632	0.7444	0.8864	0.7727	0.7239
	radoslaw-email	0.6335	0.8276	0.8198	0.6762	0.8286	0.8270	0.6679	0.8232	0.8206	0.6872	0.8264	0.8370	0.7170	0.7874	0.8593
	Eu-core	0.8751	0.8382	0.6933	0.8787	0.7658	0.7590	0.8752	0.7553	0.7907	0.8554	0.8425	0.7652	0.8866	0.8502	0.8387
	fb-forum	0.8662	0.8094	0.8004	0.8744	0.8311	0.8058	0.8645	0.8375	0.8291	0.8751	0.8100	0.7907	0.8750	0.8715	0.8160
	CollegeMsg	0.5464	0.6586	0.8707	0.5552	0.6990	0.8855	0.5357	0.6959	0.8875	0.7056	0.8604	0.8680	0.7980	0.8316	0.8776
F1 score	mit	0.8676	0.8040	0.7833	0.8889	0.7540	0.7644	0.7543	0.7861	0.7813	0.9028	0.7763	0.7353	0.8887	0.8047	0.7276
	radoslaw-email	0.0804	0.8270	0.8214	0.2968	0.8271	0.8291	0.2977	0.8297	0.8232	0.2826	0.8321	0.8385	0.4608	0.7985	0.8696
	Eu-core	0.8752	0.8425	0.6941	0.8764	0.7735	0.6153	0.8769	0.5910	0.7986	0.8570	0.8457	0.7621	0.8868	0.8554	0.8465
	fb-forum	0.8705	0.8099	0.7991	0.8748	0.8255	0.8034	0.8682	0.8393	0.8260	0.8740	0.8132	0.7883	0.8705	0.8740	0.8068
	CollegeMsg	0.5833	0.6488	0.8756	0.5883	0.7121	0.8936	0.5007	0.7407	0.8787	0.1299	0.8760	0.8686	0.1994	0.8286	0.8796

Table 8Performance comparison of proposed quantum random walk embedding with Gaussian's Naive Bayes (*QEGNB*) on different random walk techniques and different train-test ratio.

Metric	Datasets	Node2Vec	DeepWalk	Walklets	Node2Vec	DeepWalk	Walklets	Node2Vec	DeepWalk	Walklets	Node2Vec	DeepWalk	Walklets	Node2Vec	DeepWalk	Walklets
		0.5			0.6			0.7			0.8			0.9		
AUC	mit	0.8969	0.9022	0.8076	0.9245	0.8653	0.8242	0.8255	0.8771	0.7802	0.9408	0.9099	0.8194	0.8977	0.8651	0.8162
	radoslaw-email	0.4745	0.9213	0.8920	0.5991	0.9201	0.9180	0.5976	0.9224	0.8964	0.5870	0.9136	0.8959	0.6844	0.9005	0.9008
	Eu-core	0.9210	0.8545	0.6549	0.9291	0.8314	0.7330	0.9300	0.7310	0.6843	0.9095	0.8808	0.7260	0.9255	0.8822	0.8340
	fb-forum	0.9266	0.8867	0.8381	0.9302	0.8986	0.7900	0.9230	0.9082	0.7740	0.9336	0.8558	0.8276	0.9264	0.8715	0.8451
	CollegeMsg	0.6717	0.6191	0.9136	0.6901	0.7143	0.9143	0.6673	0.7154	0.9159	0.5458	0.6608	0.9172	0.6485	0.9079	0.9092
Precision	mit	0.8313	0.8204	0.7287	0.8836	0.8150	0.7346	0.6440	0.8044	0.7121	0.8370	0.8389	0.7440	0.8874	0.8150	0.7919
	radoslaw-email	0.0414	0.8548	0.8240	0.2128	0.8735	0.8356	0.1909	0.8653	0.8238	0.1873	0.8322	0.8127	0.3754	0.8459	0.8480
	Eu-core	0.8546	0.7907	0.5930	0.8711	0.7328	0.5532	0.8614	0.5203	0.7063	0.8575	0.7906	0.6469	0.8597	0.8812	0.7977
	fb-forum	0.8827	0.8136	0.7826	0.8766	0.8259	0.7263	0.8779	0.8451	0.7092	0.8605	0.8014	0.8003	0.8709	0.8840	0.7464
	CollegeMsg	0.4421	0.6438	0.8530	0.4748	0.5861	0.8532	0.3686	0.6895	0.8062	0.0815	0.6458	0.8425	0.2080	0.7890	0.8129
Recall	mit	0.8066	0.8265	0.7381	0.8589	0.7744	0.7762	0.7969	0.8105	0.7306	0.9102	0.8389	0.7507	0.8172	0.8167	0.7162
	radoslaw-email	0.3275	0.8396	0.8019	0.3239	0.8329	0.8464	0.3919	0.8475	0.8128	0.3981	0.8429	0.8359	0.4922	0.8168	0.8207
	Eu-core	0.8578	0.8043	0.6722	0.8732	0.7474	0.5639	0.8784	0.5987	0.6594	0.8543	0.8357	0.7284	0.8740	0.7417	0.6903
	fb-forum	0.8620	0.7981	0.7559	0.8734	0.8336	0.7045	0.8698	0.8152	0.6773	0.8638	0.8049	0.6751	0.8786	0.8636	0.7715
	CollegeMsg	0.8034	0.6592	0.8236	0.7697	0.6488	0.8064	0.8452	0.6939	0.8687	0.3358	0.6818	0.8552	0.5599	0.8373	0.8968
Accuracy	mit	0.8210	0.8200	0.7292	0.8667	0.7943	0.7527	0.7995	0.8035	0.7076	0.8650	0.8420	0.7704	0.8543	0.7879	0.7388
	radoslaw-email	0.6203	0.8460	0.8143	0.7363	0.8538	0.8405	0.7433	0.8552	0.8158	0.7523	0.8352	0.8245	0.7675	0.8238	0.8247
	Eu-core	0.8568	0.7844	0.6394	0.8707	0.7444	0.7364	0.8654	0.7344	0.6697	0.8558	0.8046	0.6915	0.8645	0.8180	0.7488
	fb-forum	0.8717	0.8103	0.7714	0.8750	0.8288	0.7196	0.8751	0.8322	0.6981	0.8647	0.7893	0.7370	0.8764	0.8715	0.7708
	CollegeMsg	0.5872	0.6040	0.8444	0.5850	0.6231	0.8266	0.5715	0.6171	0.8369	0.6954	0.6269	0.8350	0.6694	0.8010	0.8469
F1 score	mit	0.8187	0.8234	0.7334	0.8711	0.7942	0.7548	0.7123	0.8074	0.7213	0.8721	0.8389	0.7473	0.8508	0.8158	0.7522
	radoslaw-email	0.0736	0.8472	0.8128	0.2569	0.8527	0.8410	0.2568	0.8563	0.8183	0.2547	0.8375	0.8241	0.4259	0.8311	0.8341
	Eu-core	0.8562	0.7974	0.6301	0.8721	0.7400	0.5585	0.8699	0.5567	0.6820	0.8559	0.8125	0.6852	0.8668	0.8054	0.7401
	fb-forum	0.8722	0.8058	0.7690	0.8750	0.8297	0.7153	0.8738	0.8299	0.6929	0.8622	0.8031	0.7324	0.8747	0.8737	0.7587
	CollegeMsg	0.5703	0.6514	0.8380	0.5873	0.6159	0.8291	0.5133	0.6917	0.8363	0.1311	0.6633	0.8488	0.3034	0.8124	0.8528

machine learning classifier and Embedded Projected Quantum Kernel (*QEGNB*). These algorithms are evaluated using four assessment metrics and based on five well-known and publicly available datasets On a variety of training and testing ratios with different embedding methods, as depicted in Table 8

In terms of AUC, Node2Vec embedding is prominent over other methods for three networks. The highest scores are on the ratio of 0.8 for mit, 0.7 for Eu-core, and 0.8 for fb-forum. For radoslaw-email, Deepwalk does better and takes the highest score on ratio of 0.7. For CollegeMsg, Walklet being the consistent performer has the highest score at different ratios for different metrics. In terms of Precision, it is high for Node2Vec on mit at 0.9, Eu-core at 0.6, and fb-forum at 0.5 ratio. On radoslaw-email, Deepwalk performs best having highest on 0.6 ratio. Considering recall Node2Vec is again highest for mit and Eu-core and fb-forum with highest score on ratio of 0.8, 0.7 and 0.9 respectively. Deepwalk outperforms for radoslaw-email having highest on 0.7 ratio. For Accuracy and F1 score the score patterns are identical, here Node2Vec outperforms with highest score on ratio of 0.6 for mit, Eu-core and fb-forum with an exception that Deepwalk competes for Node2Vec with a higher score on 0.5 ratio. Deepwalk performs better for radoslaw-email having highest on 0.7/0.6 for accuracy and F1 score respectively.

5.8. Concluding performance of different embedding methods on different ML models

Looking at the performance delivered by Quantum enhanced Node2Vec/Deepwalk/Walklet embeddings on different ML models, it is evident that Node2Vec performs well on mit, Eu-core, fb-forum datasets for most of the ML models. Basically Node2Vec is the best performer amongst the three embedding generation methods as it works on biased DFS and BFS exploration. Here the three datasets mit, Eu-core and fb-forum have standard clustering coefficients like 0.7, 0.3 and 0.2 respectively, that helps the clustered and diverse exploration suitable for Node2Vec hence it gives better results. The other two i.e. radoslaw-email (4.5) and CollegeMsg (0.056) have very high or low clustering coefficients. For radoslaw-email Deepwalk that captures homophily works better and for collegemsg Walklets that captures multiscale relationships turns constantly best. After evaluating the performance of the majority of the approaches, we can conclude that, the overall performance on higher train ratios is better with the exception of the "mit" dataset at few ratios. This dataset is small, and it acts as an example of a deviation from standard performance.

5.9. Performance comparison and analysis of projected quantum kernel with combined random walk embedding methods (*PQKELP*) on different machine learning algorithms

On different training-testing ratios, this section compare and analyze the performance of combined random walk methods with Projected Quantum Kernel (*PQKELP*) on different Machine Learning algorithms. Here we merge embeddings generated from all the random walk methods i.e. Node2Vec, DeepWalk, and Walklets, to produce a rich set of features. The models have used are projected quantum embedding with Neural Network(*PQKE_NN*), with Logistic Regression (*PQKE_LR*), XGBoost(*PQKE_XGB*), Random Forest Classifier (*PQKE_RFC*), Linear Discriminant Analysis (*PQKE_LDA*), Gaussian Naive Bias (*PQKE_GNB*). show the comparison and analysis of link prediction performance on different machine learning algorithms. In order to avoid verbosity, we would refer *PQKE_NN* by *QNN*, *PQKE_RFC* by *QRFC*, *PQKE_GNB* by *QGNB* and *PQKE_XGB* by *QXGB* in this section.

In terms of AUC, *QNN*, *QRFC*, and *QGNB* are performing better for all networks i.e. *QNN* has the highest scores on ratios of 0.6, 0.7, and 0.9 for the mit network, on all ratios except 0.5 for radoslaw-email, on all ratios for Eu-core and CollegeMsg and 0.5, 0.8 for fb-forum. *QRFC* is highest on ratio of 0.5, 0.8 for mit, 0.5 for radoslaw-email, 0.6, 0.7, 0.9 for fb-forum. Lastly, *QGNB* is highest at 0.7 ratio for radoslaw-email.

For Precision, again *QNN*, *QRFC*, *QGNB* perform better and occasionally *QXGB* gives competitive results. *QNN* has the highest scores on ratios of 0.6, 0.7 for mit network, 0.6 for radoslaw-email, 0.5, 0.6, 0.7 ratios for Eu-core, 0.5 for fb-forum, and 0.5, 0.6, 0.8 for CollegeMsg. *QRFC* is highest on ratio of 0.5, 0.8 for mit, 0.5 for radoslaw-email, 0.8, 0.9 for Eu-core, 0.5, 0.6, 0.9 for fb-forum and 0.7, 0.9 for CollegeMsg. Lastly, *QGNB* is highest on a ratio of 0.9 for mit, 0.7, 0.8, and 0.9 for radoslaw-email.

Considering recall, it again has leading scores from *QNN*, *QRFC*, and *QGNB* and occasionally *QXGB* gives competitive results. *QNN* has the highest scores on a ratio of 0.9 for mit, 0.5, 0.8, 0.9 for radoslaw-email, 0.8 for fb-forum, 0.7, 0.8, 0.9 for CollegeMsg whereas for Eu-core it is highest on all ratios except 0.7. *QRFC* has exceeding scores on the ratios of 0.5, 0.6, 0.7 for mit, 0.6, 0.7 for radoslaw-email, 0.7 for Eu-core, 0.5, 0.6 for CollegeMsg whereas it is highest on Eu-core for all ratios except 0.8. *QGNB* could fetch only one highest score on 0.8 ratio for mit.

In the case of Accuracy *QNN*, *QRFC*, *QGNB*, and *QXGB* are the main competitors. *QNN* produces the best results on the ratios of 0.6

Table 9

Performance comparison of proposed projected quantum kernel on combined random walk embedding (*PQKE*) with different machine learning algorithms on different train-test ratio.

Metric	Dataset	Ratio	<i>PQKE_NN</i>	<i>PQKE_LR</i>	<i>PQKE_XGB</i>	<i>PQKE_RFC</i>	<i>PQKE_LDA</i>	<i>PQKE_GNB</i>
AUC	mit	0.5	0.865309278	0.623328	0.722145	0.873185	0.689314	0.825693
		0.6	0.864695	0.62771	0.701762	0.841455	0.616961	0.810102
		0.7	0.860858	0.649041	0.721625	0.853817	0.649304	0.848906
		0.8	0.821085	0.615558	0.70528	0.849478	0.615711	0.827142
		0.9	0.847872	0.61947	0.669262	0.82619	0.63396	0.84469
	radoslaw-email	0.5	0.894978	0.690306	0.772632	0.907954	0.690827	0.904418
		0.6	0.90589	0.690901	0.749885	0.902033	0.692007	0.891812
		0.7	0.916782	0.704864	0.783494	0.917257	0.705038	0.919613
		0.8	0.911493	0.711007	0.768137	0.907358	0.712417	0.900925
		0.9	0.916062	0.701747	0.761836	0.906678	0.69843	0.909006
	Eu-core	0.5	0.883062	0.791341	0.769906	0.872045	0.789268	0.832581
		0.6	0.896575	0.802742	0.779729	0.89467	0.802624	0.848307
		0.7	0.886766	0.789495	0.77837	0.883679	0.788101	0.841354
		0.8	0.877467	0.782154	0.778161	0.871876	0.781536	0.833984
		0.9	0.859335	0.643819	0.719521	0.790471	0.642978	0.685467
	fb-forum	0.5	0.8553966	0.69369	0.753322	0.875808	0.689314	0.80866
		0.6	0.853646	0.643459	0.745053	0.868097	0.644388	0.7678
		0.7	0.860507	0.675676	0.739316	0.877166	0.674599	0.775907
		0.8	0.855355	0.685995	0.71624	0.816075	0.685387	0.788441
		0.9	0.857219	0.683821	0.777266	0.879906	0.679434	0.780536
	CollegeMsg	0.5	0.940414409	0.918061	0.850709	0.931198	0.918352	0.92215
		0.6	0.941398	0.921469	0.872028	0.937568	0.920372	0.925605
		0.7	0.954796	0.945458	0.893932	0.937393	0.942719	0.948143
		0.8	0.954636	0.944915	0.903973	0.950123	0.942929	0.9402
		0.9	0.945508	0.936235	0.896798	0.9311	0.933862	0.924193
Precision	mit	0.5	0.805128137	0.601056	0.718312	0.805844	0.598337	0.755749
		0.6	0.774848	0.573154	0.686048	0.763329	0.557666	0.714871
		0.7	0.778489	0.573658	0.693845	0.741471	0.576955	0.760737
		0.8	0.762917	0.614327	0.736844	0.814764	0.618697	0.74804
		0.9	0.75781	0.643995	0.736795	0.812903	0.642361	0.820746
	radoslaw-email	0.5	0.794054	0.616876	0.733406	0.817197	0.615458	0.804022
		0.6	0.830829	0.637601	0.771926	0.823235	0.642125	0.808
		0.7	0.830778	0.644146	0.792452	0.832893	0.644973	0.848963
		0.8	0.832678	0.675299	0.766605	0.839396	0.671313	0.842202
		0.9	0.780694	0.619501	0.7625	0.795524	0.616248	0.829064
	Eu-core	0.5	0.828822	0.733354	0.769244	0.842492	0.732608	0.784463
		0.6	0.828639	0.724923	0.784772	0.82412	0.722694	0.744048
		0.7	0.857891	0.725927	0.802143	0.839507	0.726542	0.74537
		0.8	0.794407	0.690107	0.751966	0.810194	0.695277	0.737448
		0.9	0.673316	0.52601	0.612659	0.678343	0.532675	0.538615
	fb-forum	0.5	0.807781955	0.638876	0.781315	0.791304	0.638554	0.718384
		0.6	0.773318	0.599247	0.722213	0.789598	0.593904	0.697214
		0.7	0.818913	0.62258	0.717398	0.806941	0.632324	0.704202
		0.8	0.751083	0.648896	0.696682	0.763151	0.650394	0.722611
		0.9	0.844877	0.635701	0.81346	0.817765	0.643519	0.71951
	CollegeMsg	0.5	0.893164337	0.849051	0.833744	0.847391	0.847492	0.838845
		0.6	0.901876	0.856392	0.867095	0.852304	0.859574	0.851355
		0.7	0.884154	0.884118	0.874442	0.897049	0.888283	0.888655
		0.8	0.89445	0.880801	0.897104	0.89821	0.877342	0.858058
		0.9	0.869431	0.885432	0.86103	0.891355	0.864737	0.850011

(continued on next page)

and 0.7 for mit. For Eu-core and CollegeMsg it has the highest scores on all the ratios and for fb-forum is best at 0.8. *QRFC* is best on ratios of 0.5, 0.8 for mit, 0.5, 0.6, 0.8 for radoslaw-email, 0.8 for Eu-core, and 0.6 for CollegeMsg whereas it is highest for fb-forum on all ratios. *QXGB* produces best results for a few places including 0.9 for mit and 0.7, 0.9 for radoslaw-email. For the F1 score, the model's performance is similar to Accuracy with minor deviations on some ratios.

Comparing the performance of all these combined models it is strikingly evident that *QNN*, *QRFC*, and *QGNB* are highest highest-performing models with some competitive scores shared by *QXGB*.

QLR and *QLDA* perform less when compared with the stated exceeding models.

5.10. Performance comparison of proposed projected quantum kernel with combined random walk embedding methods (*PQKELP*) with state-of-the-art methods

This section compares the performance of four state-of-the-art approaches with some of the best-performing models of our proposed approach. Note that these models use the combined embeddings generated from all the random walk methods similar to the previous section.

Table 9 (continued).

Metric	Dataset	Ratio	<i>PQKE_NN</i>	<i>PQKE_LR</i>	<i>PQKE_XGB</i>	<i>PQKE_RFC</i>	<i>PQKE_LDA</i>	<i>PQKE_GNB</i>
Recall	mit	0.5	0.7528162	0.58866	0.748084	0.777549	0.60331	0.711068
		0.6	0.767626	0.614247	0.700092	0.77731	0.595359	0.720764
		0.7	0.788144	0.63942	0.741242	0.796363	0.630774	0.792816
		0.8	0.738632	0.56873	0.70268	0.74621	0.587526	0.765146
		0.9	0.840913	0.567238	0.649097	0.685028	0.576247	0.739188
	radoslaw-email	0.5	0.858906	0.632448	0.822535	0.82818	0.636025	0.842602
		0.6	0.830829	0.637601	0.771926	0.823235	0.642125	0.808
		0.7	0.857183	0.649099	0.769944	0.861242	0.651724	0.853265
		0.8	0.869103	0.644836	0.794434	0.838328	0.645913	0.829049
		0.9	0.888822	0.677614	0.734055	0.821297	0.672369	0.856183
	Eu-core	0.5	0.841973	0.676644	0.780467	0.770396	0.671811	0.635247
		0.6	0.845151	0.73019	0.783913	0.844726	0.73043	0.769895
		0.7	0.819928	0.739814	0.761263	0.844167	0.73397	0.766881
		0.8	0.847906	0.689364	0.792174	0.800217	0.690852	0.702272
		0.9	0.828689	0.594562	0.788916	0.783956	0.60579	0.517316
	fb-forum	0.5	0.745757739	0.636232	0.711054	0.82183	0.631617	0.761045
		0.6	0.751123	0.605823	0.797422	0.818645	0.599171	0.734633
		0.7	0.760962	0.615252	0.781349	0.785384	0.62451	0.682796
		0.8	0.801809	0.591997	0.758361	0.725364	0.601275	0.677106
		0.9	0.675772	0.624481	0.740108	0.795399	0.615457	0.770266
	CollegeMsg	0.5	0.886580626	0.857339	0.85722	0.906438	0.84873	0.842953
		0.6	0.836176	0.841788	0.881277	0.916355	0.841905	0.845518
		0.7	0.943318	0.924542	0.905351	0.925306	0.92416	0.924701
		0.8	0.941601	0.921055	0.898633	0.9305	0.915103	0.881596
		0.9	0.978971	0.927789	0.952117	0.933599	0.933614	0.940498

Considering the exceeding performance from the previous section four models namely projected quantum embedding with Neural Network (*QNN*), XGBoost (*QXGB*), random forest classifier (*QRFC*), and Gaussian Naive Bayes (*QGNB*), they are chosen for comparison. show the performance of these selected models with state-of-the-art methods. The performance of all of these algorithms is compared using five assessment metrics: AUC, Precision, Recall, Accuracy, and F1 score. The results are all based on five well-known and publicly available dynamic datasets on different training testing ratios. Hence a four-dimensional comparison is presented here.

To avoid verbosity, we will be referring *PQKE_NN*, *PQKE_XGB*, *PQKE_RFC*, *PQKE_GNB* with *QNN*, *QXGB*, *QRFC*, *QGNB* respectively. In terms of AUC, for the mit network, *QNN* outperforms other models as well as state-of-the-art methods on ratios of 0.6, 0.7, and 0.9, and *QRFC* is best on 0.5 and 0.8. For radoslaw-email, *QNN* outperforms on ratios of 0.6, 0.8, and 0.9, and *QRFC* is best on 0.5, and *QGNB* works for 0.7. For Eu-core and CollegeMsg *QNN* produces the highest for all ratios and *QRFC* gets a competitive score on a few ratios. For fb-forum, *QRFC* gives the best results for all ratios except for 0.8 where *QNN* gets the best score. For Precision, in the case of mit *QNN* scores highest on ratios of 0.6, 0.7, *QRFC* scores best on 0.5 and 0.8, and *QGNB* on 0.9. For radoslaw-email, *QGNB* is highest on 0.7, 0.8, 0.9 and *QRFC* high on 0.5 *QNN* highest on 0.6 ratio. Eu-core has *QRFC* outperformed on ratios of 0.5, 0.8, 0.9 leaving 0.6 and 0.7 for *QNN*. For fb-forum, *QNN* is best on 0.5, 0.7, 0.9, and *QRFC* on 0.6 and 0.8. CollegeMsg has *QRFC* outperformed on 0.7, 0.8, 0.9 leaving top two ratios for *QNN*.

Considering recall, mit has *QRFC* with the highest score on ratios of 0.5, 0.6, 0.7, and *QNN* on 0.9 and *QGNB* on 0.8. For radoslaw-email and Eu-core, we have *QNN* outperform on all ratios except for 0.7 where *QRFC* is the top scorer. The reverse case with fb-forum. For CollegeMsg we have *QRFC* producing top scores on ratios of 0.5, 0.6, and 0.8 and *QNN* exceeding on 0.7 and 0.9. In the case of accuracy, for mit network, it is spread across methods where our four models achieve higher scores for most of the ratios but marginally lag by LGQ on ratios of 0.7 and 0.9. For radoslaw-email, *QNN* gets highest on 0.8, *QRFC* on 0.5, 0.6, and *QGNB* on 0.7 and 0.9 ratio. For Eu-core again our quantum models outperform but lag marginally by LGQ on a ratio

of 0.9. A similar case for fb-forum LGQ exceeds marginally on 0.8 and 0.9. CollegeMsg is dominated by *QNN* on all ratios except 0.6 where *QRFC* gets the best score.

Lastly for F1 Score, when consider the mit network, *QNN* is highest on ratios of 0.6, 0.7, and 0.9 and *QRFC* on 0.5 and 0.8. radoslaw-email has *QRFC* best on 0.5, 0.6 and *QGNB* on 0.7 and 0.9, *QRFC* on 0.8 ratio. For Eu-core we have *QNN* exceeding on all ratios except for 0.7 with *QRFC*. For fb-forum, *QRFC* is the winner on all ratios except 0.8 with *QNN*. For CollegeMsg *QRFC* is highest on ratios of 0.6, 0.7, 0.8 leaving 0.5 and 0.9 highest for *QNN*.

Comparing the performance of all the metrics on different train-test ratios. It is strikingly evident that our models outperform the state-of-the-art methods. Specifically, *QNN* and *QRFC* perform extremely well. The classical equivalents of this method Node2Vec, DeepWalk, Walklets when utilized with classical machine learning models is presented by Vital, Silva, and Amancio (2024). In Singh, Kumar, and Biswas (2024), very fine comparison of Neural Network and Support Vector Machines with their Quantum counterparts i.e. QNN and QVC is presented. For the task of link prediction in dynamic network several niche methods exist that work with classical machine learning (Kumar, Mishra, & Biswas, 2022b), (Kumar, Mishra, & Biswas, 2022a; Kumar, Mishra, Pandey, & Biswas, 2022).

5.11. Complexity analysis

Having two-step feature enhancement may look like an overhead but looking at the methodology in-depth, this may appear trivial because the quantum projection is done with the famous kernel trick i.e. with simple kernel functions the features are easily separated out in high dimensional space. This is done to relabel and project the data back to the classical state, hence the computational overhead by pure quantum computation is eradicated and the learning takes place with the classical ML model. Eventually, our approach does not add significant overhead to the popular random walk embedding-based link prediction methods. To represent Big-O notation for the complexity, let us consider 'N' as the number of nodes, 'L' length of the walk, 'T' number of walks, 'W' as the window size, and 'D' dimensionality of embeddings for the Word2Vec skip-gram model. Now the random walk

Table 10

Performance comparison of proposed projected quantum kernel on combined random walk embedding (*PQKE*) with different machine learning algorithms on different train–test ratio.

Metric	Dataset	Ratio	<i>PQKE_NN</i>	<i>PQKE_LR</i>	<i>PQKE_XGB</i>	<i>PQKE_RFC</i>	<i>PQKE_LDA</i>	<i>PQKE_GNB</i>
Accuracy	mit	0.5	0.780235867	0.594536	0.720316	0.79217	0.594551	0.738223
		0.6	0.779676	0.597859	0.702458	0.778446	0.581657	0.729785
		0.7	0.789213	0.603071	0.720969	0.770804	0.604729	0.782555
		0.8	0.742938	0.580095	0.707418	0.774774	0.587577	0.737533
		0.9	0.746309	0.578866	0.66519	0.730925	0.578866	0.761305
	radoslaw-email	0.5	0.77746	0.637048	0.696648	0.855007	0.645792	0.831312
		0.6	0.825496	0.635915	0.750905	0.827446	0.639421	0.810422
		0.7	0.838455	0.645679	0.783618	0.844058	0.646966	0.850907
		0.8	0.839302	0.661819	0.767071	0.836731	0.659895	0.833545
		0.9	0.82563	0.6485	0.763488	0.813891	0.644644	0.847448
	Eu-core	0.5	0.830628	0.71149	0.769374	0.809797	0.709649	0.727412
		0.6	0.830651	0.722605	0.77931	0.829119	0.721073	0.748276
		0.7	0.835378	0.719836	0.777096	0.834867	0.718303	0.738241
		0.8	0.810497	0.706865	0.776715	0.818152	0.710705	0.742176
		0.9	0.757296	0.609831	0.706605	0.752688	0.615975	0.617512
	fb-forum	0.5	0.782725692	0.63849	0.752871	0.802481	0.637115	0.73038
		0.6	0.757471	0.598851	0.745402	0.801724	0.593103	0.708621
		0.7	0.797543	0.624262	0.738521	0.799889	0.633469	0.698628
		0.8	0.771626	0.637832	0.71511	0.752018	0.641292	0.713956
		0.9	0.761574	0.613426	0.773148	0.798611	0.618056	0.719907
	CollegeMsg	0.5	0.895558814	0.859176	0.850411	0.875343	0.855134	0.847711
		0.6	0.874479	0.851724	0.872772	0.878677	0.85341	0.849192
		0.7	0.914319	0.906417	0.8929	0.913193	0.908673	0.908665
		0.8	0.922166	0.903553	0.905245	0.918782	0.898477	0.874788
		0.9	0.914966	0.901361	0.897959	0.908163	0.891156	0.884354
F1 score	mit	0.5	0.778093916	0.594793	0.732896	0.791443	0.600813	0.732728
		0.6	0.77122	0.592989	0.692999	0.770256	0.575897	0.717805
		0.7	0.783287	0.604756	0.71676	0.767937	0.602665	0.776446
		0.8	0.750578	0.59065	0.719357	0.778982	0.602709	0.756496
		0.9	0.797202	0.603184	0.690172	0.743507	0.60751	0.777835
	radoslaw-email	0.5	0.816156	0.63474	0.754376	0.84138	0.640871	0.836919
		0.6	0.826445	0.641332	0.745609	0.830992	0.64343	0.813273
		0.7	0.843774	0.646613	0.781036	0.84683	0.648331	0.851108
		0.8	0.850501	0.659716	0.780272	0.838862	0.658368	0.835574
		0.9	0.831257	0.647256	0.748007	0.808205	0.643086	0.842405
	Eu-core	0.5	0.835345	0.703859	0.774815	0.804832	0.700894	0.702014
		0.6	0.836813	0.727547	0.784342	0.834296	0.726541	0.756751
		0.7	0.83848	0.732804	0.781168	0.84183	0.730237	0.755972
		0.8	0.820285	0.689735	0.771546	0.805175	0.693058	0.719431
		0.9	0.742966	0.558189	0.689705	0.727335	0.566885	0.527751
	fb-forum	0.5	0.775531705	0.637551	0.744531	0.806278	0.635066	0.7391
		0.6	0.762059	0.602517	0.757956	0.803859	0.596526	0.715434
		0.7	0.788875	0.618894	0.748009	0.796016	0.628392	0.693334
		0.8	0.775617	0.619142	0.726214	0.743778	0.624871	0.699119
		0.9	0.750922	0.630041	0.775052	0.806427	0.629175	0.744023
	CollegeMsg	0.5	0.889860304	0.853175	0.845319	0.875921	0.848111	0.840894
		0.6	0.867784	0.849027	0.874129	0.883169	0.850648	0.848427
		0.7	0.912778	0.903878	0.889628	0.910959	0.905867	0.90632
		0.8	0.91742	0.900478	0.897868	0.91407	0.895825	0.869668
		0.9	0.920955	0.906116	0.904285	0.911988	0.897857	0.892968

embedding-based approach involves two phases, the first one is the random walk generation phase having the complexity of order $O(N * T * L)$ and the second one is embedding generation with the Word2Vec skip-gram model which has the complexity of order $O(N * D * W)$. Considering the window size and number of walks small and constant, we may broadly say that it has the complexity of $O(N^2)$. Now, for computation of kernel matrix with Gaussian kernel function the complexity would be $O(N^2 * D)$. For larger networks $N \gg D$, so, we may again broadly say that the overall model complexity does not go much beyond $O(N^2)$. Summing up the overall complexity of random walk embedding and Quantum Projection it gives complexity of $O(2XN^2)$ which incurs only linear overhead in terms of computation time.

6. Conclusion

This work aimed to investigate the impact of quantum projection on the well-known random walk embedding-based link prediction approach. To address this, we suggested the *PQKELP* model, which uses a projected quantum kernel (PQK) on random walk-based embedded features and produces quantum-enhanced embedded features resulting in improved link prediction performance. Utilizing various embedding techniques and machine learning models, we demonstrate that our *PQKELP* produces remarkable outcomes. We have created the following models: Gaussian Naive Bias based *QE_GNB*, Random

Table 11

Performance comparison of proposed projected quantum kernel with combined random walk embedding based link prediction framework (*PQKELP*) with state-of-the-art methods on different train–test ratio.

Metric	Dataset	Ratio	WEAK	LGQ	CTDNE	<i>PQKE_XGB</i>	<i>PQKE_NN</i>	<i>PQKE_RFC</i>	<i>PQKE_GNB</i>
AUC	mit	0.5	0.617	0.74033	0.75271	0.722145	0.865309278	0.873185	0.825693
		0.6	0.66092	0.75543	0.76134	0.701762	0.864695	0.841455	0.810102
		0.7	0.63901	0.70946	0.75291	0.721625	0.860858	0.853817	0.848906
		0.8	0.67299	0.77156	0.75994	0.70528	0.821085	0.849478	0.827142
		0.9	0.64556	0.75329	0.74519	0.669262	0.847872	0.82619	0.84469
	radoslaw-email	0.5	0.65295	0.7208	0.76217	0.772632	0.894978	0.907954	0.904418
		0.6	0.6528	0.72555	0.75998	0.749885	0.90589	0.902033	0.891812
		0.7	0.65315	0.7782	0.76232	0.783494	0.916782	0.917257	0.919613
		0.8	0.65705	0.77224	0.76188	0.768137	0.911493	0.907358	0.900925
		0.9	0.66052	0.74361	0.76509	0.761836	0.916062	0.906678	0.909006
	Eu-core	0.5	0.6351	0.69402	0.73793	0.769906	0.883062	0.872045	0.832581
		0.6	0.67294	0.72448	0.74588	0.779729	0.896575	0.89467	0.848307
		0.7	0.65241	0.74385	0.7459	0.77837	0.886766	0.883679	0.841354
		0.8	0.71175	0.72803	0.74398	0.778161	0.877467	0.871876	0.833984
		0.9	0.6961	0.73109	0.75201	0.719521	0.859335	0.790471	0.685467
	fb-forum	0.5	0.61658	0.56263	0.67884	0.753322	0.8553966	0.875808	0.80866
		0.6	0.63759	0.60912	0.67527	0.745053	0.853646	0.868097	0.7678
		0.7	0.62829	0.61618	0.67444	0.739316	0.860507	0.877166	0.775907
		0.8	0.64071	0.62719	0.67739	0.71624	0.855355	0.816075	0.788441
		0.9	0.63201	0.62178	0.68133	0.777266	0.857219	0.879906	0.780536
	CollegeMsg	0.5	0.5499	0.5122	0.60082	0.850709	0.940414409	0.931198	0.92215
		0.6	0.55904	0.50369	0.59807	0.872028	0.941398	0.937568	0.925605
		0.7	0.56788	0.51324	0.59877	0.893932	0.954796	0.937393	0.948143
		0.8	0.56846	0.52384	0.61921	0.903973	0.954636	0.950123	0.9402
		0.9	0.58615	0.51936	0.61423	0.896798	0.945508	0.9311	0.924193

(continued on next page)

Forest Classifier based *QE_RFC*, Linear Regression based *QE_LR*, Linear Discriminant Analysis model based *QE_LDA*, and Neural Network based *QE_NN*. All these are evaluated on three embedding methods as described in Section 4.1, Node2Vec, Deepwalk, and Walklets. Furthermore, we developed six additional combined models based on embeddings from all the aforementioned embedding methods and varying over all of the previously mentioned ML models, termed as *PQKE_**. Using five standard datasets, we tested with various train–test ratios for these model variations and assessed their performance on five performance criteria. Higher train ratios, such as 0.8 and 0.9, have shown improved performance for the models, and some ML models – such as the Random Forest Classifier, Neural Network, and Gaussian Naive Bias – deliver remarkable outcomes. Our models outperform the conventional models – WEAK, LGQ, and CTDNE – when compared to state-of-the-art techniques. Ultimately, we deduced that this is accomplished with minimal additional computational expense.

7. Limitations and future work

Being in the Noisy Intermediate-Scale Quantum (NISQ) era, the current quantum hardware has limited qubits, noisy and prone to error. With large number of qubits it is difficult to control and maintain coherence. The proposed approach of quantum projection through kernel is efficient, still, it faces the limitation of less number of qubits on simulation while projecting the huge dimensions of the embedded features into quantum space. When we integrate features from all three embedding approaches, this becomes much more crucial. As a result, we must employ dimensionality reduction techniques, which cause performance to slightly decline.

Subsequent research in this area may focus on solving the projection issue by either executing it on real quantum machines with a higher qubit count or by using enhanced simulators that are capable of running in parallel (Häner & Steiger, 2017). The production of weight-biased embedded features (Riascos & Mateos, 2021), which show additional node and network properties, could be another customization for this. This method can be further applied to link prediction in attributed

and multiplex networks, among other types of networks. Precisely, the quantum enhanced node embeddings achieved through PQKELP could benefit several other dynamic network tasks like analyzing proteins–protein interaction and their roles in disease mechanisms for drug discovery (Parenti & Rastelli, 2012), traffic congestion prediction (Qu, Liu, & Zheng, 2022), for traffic flow prediction in iot-based networks (Yan, Wang, Yu, Jin, & Zhang, 2021), for efficient analysis of cancer cell interactions in disease prediction (Pati et al., 2022), in quantum sensor networks (Qian et al., 2019).

CRedit authorship contribution statement

Mukesh Kumar: Conceptualization, Methodology, Writing – review & editing, Principal Supervisor. **Nisha Singh:** Conceptualization, Methodology, Software, Writing – original draft, Writing – review & editing. **Bhaskar Biswas:** Investigation, Visualization, Supervision.

Declaration of competing interest

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

Acknowledgments

The support and resources provided by PARAM Shivay Facility under the National Supercomputing Mission, Government of India, at the Indian Institute of Technology, Varanasi, Uttar Pradesh, India, are gratefully acknowledged. I also thank to Kalinga Institute of Industrial Technology (KIIT), Deemed to be University, Bhubaneswar, Odisha, India for continuous support.

Data availability

The research data has been provided

Dynamic Dataset (Reference data) (Mendeley Data)

Table 11 (continued).

Metric	Dataset	Ratio	WEAK	LGQ	CTDNE	<i>PQKE_XGB</i>	<i>PQKE_NN</i>	<i>PQKE_RFC</i>	<i>PQKE_GNB</i>
Precision	mit	0.5	0.54657	0.5028	0.5212	0.718312	0.805128137	0.805844	0.755749
		0.6	0.51832	0.59326	0.56623	0.686048	0.774848	0.763329	0.714871
		0.7	0.4245	0.52969	0.5462	0.693845	0.778489	0.741471	0.760737
		0.8	0.56347	0.50243	0.51422	0.736844	0.762917	0.814764	0.74804
		0.9	0.55564	0.50821	0.47396	0.736795	0.75781	0.812903	0.820746
	radoslaw-email	0.5	0.35189	0.58129	0.02322	0.733406	0.794054	0.817197	0.804022
		0.6	0.34276	0.55966	0.36698	0.771926	0.830829	0.823235	0.808
		0.7	0.39013	0.66113	0.03082	0.792452	0.830778	0.832893	0.848963
		0.8	0.40074	0.59591	0.36698	0.766605	0.832678	0.839396	0.842202
		0.9	0.36674	0.57315	0.36698	0.7625	0.780694	0.795524	0.829064
	Eu-core	0.5	0.46016	0.64236	0.77165	0.769244	0.828822	0.842492	0.784463
		0.6	0.5181	0.63464	0.74187	0.784772	0.828639	0.82412	0.744048
		0.7	0.59766	0.55555	0.77747	0.802143	0.857891	0.839507	0.74537
		0.8	0.49642	0.62853	0.76441	0.751966	0.794407	0.810194	0.737448
		0.9	0.57674	0.61669	0.71007	0.612659	0.673316	0.678343	0.538615
	fb-forum	0.5	0.51143	0.58032	0.29523	0.781315	0.807781955	0.791304	0.718384
		0.6	0.46522	0.49895	0.29523	0.722213	0.773318	0.789598	0.697214
		0.7	0.51716	0.58204	0.29523	0.717398	0.818913	0.806941	0.704202
		0.8	0.5513	0.57903	0.29523	0.696682	0.751083	0.763151	0.722611
		0.9	0.61347	0.60902	0.29523	0.81346	0.844877	0.817765	0.71951
	CollegeMsg	0.5	0.29069	0.11573	0.32706	0.833744	0.893164337	0.847391	0.838845
		0.6	0.36475	0.24242	0.51053	0.867095	0.901876	0.852304	0.851355
		0.7	0.37437	0.20857	0.4567	0.874442	0.884154	0.897049	0.888655
		0.8	0.37342	0.23206	0.58425	0.897104	0.89445	0.89821	0.858058
		0.9	0.41565	0.24565	0.46667	0.86103	0.869431	0.891355	0.850011
Recall	mit	0.5	0.401	0.65248	0.18702	0.748084	0.7528162	0.777549	0.711068
		0.6	0.51832	0.59326	0.18836	0.700092	0.767626	0.77731	0.720764
		0.7	0.52048	0.46957	0.22178	0.741242	0.788144	0.796363	0.792816
		0.8	0.47348	0.7054	0.21071	0.70268	0.738632	0.74621	0.765146
		0.9	0.4833	0.54416	0.19292	0.649097	0.840913	0.685028	0.739188
	radoslaw-email	0.5	0.52715	0.43379	0.00793	0.822535	0.858906	0.82818	0.842602
		0.6	0.34276	0.55966	0.74332	0.771926	0.830829	0.823235	0.808
		0.7	0.49617	0.46873	0.00507	0.769944	0.857183	0.861242	0.853265
		0.8	0.51376	0.49058	0.74332	0.794434	0.869103	0.838328	0.829049
		0.9	0.48106	0.52094	0.00034	0.734055	0.888822	0.821297	0.856183
	Eu-core	0.5	0.42983	0.45085	0.09737	0.780467	0.841973	0.770396	0.635247
		0.6	0.5181	0.63464	0.11177	0.783913	0.845151	0.844726	0.769895
		0.7	0.35609	0.59309	0.10959	0.761263	0.819928	0.844167	0.766881
		0.8	0.53407	0.52227	0.09868	0.792174	0.847906	0.800217	0.702272
		0.9	0.47652	0.52594	0.12693	0.788916	0.828689	0.783956	0.517316
	fb-forum	0.5	0.32328	0.18157	0.34139	0.711054	0.745757739	0.82183	0.761045
		0.6	0.46522	0.49895	0.60366	0.797422	0.751123	0.818645	0.734633
		0.7	0.35221	0.28503	0.60366	0.781349	0.760962	0.785384	0.682796
		0.8	0.33666	0.3224	0.60366	0.758361	0.801809	0.725364	0.677106
		0.9	0.27195	0.26203	0.60366	0.740108	0.675772	0.795399	0.770266
	CollegeMsg	0.5	0.20712	0.13945	0.01824	0.85722	0.886580626	0.906438	0.842953
		0.6	0.36475	0.24242	0.01187	0.881277	0.836176	0.916355	0.845518
		0.7	0.1954	0.0857	0.02664	0.905351	0.943318	0.925306	0.924701
		0.8	0.19295	0.09989	0.04149	0.898633	0.941601	0.9305	0.881596
		0.9	0.21831	0.09629	0.02485	0.952117	0.978971	0.933599	0.940498

Table 12

Performance comparison of proposed projected quantum kernel on random walk embedding based link Prediction framework (*PQKELP*) with state-of-the-art methods on different train-test ratio.

Metric	Dataset	Ratio	WEAK	LGQ	CTDNE	<i>PQKE_XGB</i>	<i>PQKE_NN</i>	<i>PQKE_RFC</i>	<i>PQKE_GNB</i>
Accuracy	mit	0.5	0.72901	0.73379	0.6772	0.720316	0.780235867	0.79217	0.738223
		0.6	0.71635	0.79454	0.69545	0.702458	0.779676	0.778446	0.729785
		0.7	0.69137	0.79554	0.67856	0.720969	0.789213	0.770804	0.782555
		0.8	0.77516	0.7425	0.6842	0.707418	0.742938	0.774774	0.737533
		0.9	0.73492	0.77929	0.67506	0.66519	0.746309	0.730925	0.761305
	radoslaw-email	0.5	0.69659	0.78237	0.619	0.696648	0.77746	0.855007	0.831312
		0.6	0.70988	0.75176	0.62691	0.750905	0.825496	0.827446	0.810422
		0.7	0.70298	0.80294	0.61993	0.783618	0.838455	0.844058	0.850907
		0.8	0.69857	0.78197	0.62347	0.767071	0.839302	0.836731	0.833545
		0.9	0.71739	0.747	0.62667	0.763488	0.82563	0.813891	0.847448
	Eu-core	0.5	0.74501	0.83203	0.64456	0.769374	0.830628	0.809797	0.727412
		0.6	0.77783	0.82736	0.67003	0.77931	0.830651	0.829119	0.748276
		0.7	0.81342	0.80293	0.65057	0.777096	0.835378	0.834867	0.738241
		0.8	0.78945	0.8146	0.65376	0.776715	0.810497	0.818152	0.742176
		0.9	0.79009	0.81177	0.68905	0.706605	0.757296	0.752688	0.617512
	fb-forum	0.5	0.76111	0.78237	0.46523	0.752871	0.782725692	0.802481	0.73038
		0.6	0.68578	0.75164	0.46603	0.745402	0.757471	0.801724	0.708621
		0.7	0.75996	0.78644	0.47329	0.738521	0.797543	0.799889	0.698628
		0.8	0.76923	0.78559	0.47466	0.71511	0.771626	0.752018	0.713956
		0.9	0.78383	0.80042	0.47009	0.773148	0.761574	0.798611	0.719907
	CollegeMsg	0.5	0.62698	0.61408	0.64281	0.850411	0.895558814	0.875343	0.847711
		0.6	0.60357	0.62299	0.63544	0.872772	0.874479	0.878677	0.849192
		0.7	0.62424	0.62685	0.63731	0.8929	0.914319	0.913193	0.908665
		0.8	0.61518	0.63122	0.6361	0.905245	0.922166	0.918782	0.874788
		0.9	0.656	0.62825	0.64548	0.897959	0.914966	0.908163	0.884354
F1 score	mit	0.5	0.32706	0.52934	0.49182	0.732896	0.778093916	0.791443	0.732728
		0.6	0.4188	0.53716	0.50949	0.692999	0.77122	0.770256	0.717805
		0.7	0.37858	0.47527	0.48783	0.71676	0.783287	0.767937	0.776446
		0.8	0.43814	0.56696	0.49619	0.719357	0.750578	0.778982	0.756496
		0.9	0.41643	0.50684	0.47775	0.690172	0.797202	0.743507	0.777835
	radoslaw-email	0.5	0.35074	0.44353	0.46391	0.754376	0.816156	0.84138	0.836919
		0.6	0.343	0.45113	0.46489	0.745609	0.826445	0.830992	0.813273
		0.7	0.3527	0.54245	0.46447	0.781036	0.843774	0.84683	0.851108
		0.8	0.3444	0.51782	0.46269	0.780272	0.850501	0.838862	0.835574
		0.9	0.35366	0.50341	0.47038	0.748007	0.831257	0.808205	0.842405
	Eu-core	0.5	0.31598	0.48087	0.40273	0.774815	0.835345	0.804832	0.702014
		0.6	0.37342	0.50559	0.41708	0.784342	0.836813	0.834296	0.756751
		0.7	0.37652	0.51163	0.40787	0.781168	0.83848	0.84183	0.755972
		0.8	0.43337	0.49483	0.41269	0.771546	0.820285	0.805175	0.719431
		0.9	0.41356	0.47086	0.42713	0.689705	0.742966	0.727335	0.527751
	fb-forum	0.5	0.30839	0.23364	0.33259	0.744531	0.775531705	0.806278	0.7391
		0.6	0.34576	0.32212	0.33198	0.757956	0.762059	0.803859	0.715434
		0.7	0.32829	0.32013	0.33957	0.748009	0.788875	0.796016	0.693334
		0.8	0.34597	0.34011	0.33205	0.726214	0.775617	0.743778	0.699119
		0.9	0.33312	0.31685	0.33495	0.775052	0.750922	0.806427	0.744023
	CollegeMsg	0.5	0.26197	0.23245	0.30841	0.845319	0.889860304	0.875921	0.840894
		0.6	0.27847	0.2191	0.31341	0.874129	0.867784	0.883169	0.848427
		0.7	0.28556	0.22458	0.30904	0.889628	0.912778	0.910959	0.90632
		0.8	0.27923	0.22926	0.32026	0.897868	0.91742	0.91407	0.869668
		0.9	0.30385	0.24057	0.30576	0.904285	0.920955	0.911988	0.892968

References

- Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., et al. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. URL: <https://www.tensorflow.org/>. Software available from tensorflow.org.
- Adafre, S. F., & de Rijke, M. (2005). Discovering missing links in wikipedia. In *Proceedings of the 3rd international workshop on link discovery* (pp. 90–97). <http://dx.doi.org/10.1145/1134271.1134284>.
- 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* (pp. 50–59).
- Aharonov, Y., Davidovich, L., & Zagury, N. (1993). Quantum random walks. *Physical Review A*, 48(2), 1687–1690. URL: www.scopus.com. Cited By :1151.
- Ahmed, N. M., Chen, L., Wang, Y., Li, B., Li, Y., & Liu, W. (2018). DeepEye: Link prediction in dynamic networks based on non-negative matrix factorization. *Big Data Mining and Analytics*, 1(1), 19–33.
- Airolidi, E. M., Blei, D. M., Fienberg, S. E., Xing, E. P., & Jaakkola, T. (2006). Mixed membership stochastic block models for relational data with application to protein-protein interactions. In *Proceedings of the international biometrics society annual meeting*, vol. 15.
- Al Hasan, M., & Zaki, M. J. (2011). A survey of link prediction in social networks. In *Social network data analytics* (pp. 243–275). Springer.
- Amara, A., Taieb, M. A. H., & Aouicha, M. B. (2021). Network representation learning systematic review: Ancestors and current development state. *Machine Learning with Applications*, 6, Article 100130.
- Anguita, D., Ridella, S., Riveccio, F., & Zunino, R. (2003). Quantum optimization for training support vector machines. *Neural Networks*, 16(5–6), 763–770.
- Arunachalam, S., & de Wolf, R. (2017). Guest column: A survey of quantum learning theory. *ACM SIGACT News*, 48(2), 41–67.
- Ashwin, N., & Ashvin, V. (2000). *Quantum walk on the line*. Center for Discrete Mathematics & Theoretical Computer Science.
- Atci, A., & Servodio, R. A. (2007). Quantum algorithms for learning and testing juntas. *Quantum Information Processing*, 6(5), 323–348.
- Attal, S., Petruccione, F., Sabot, C., & Sinayskiy, I. (2012). Open quantum random walks. *Journal of Statistical Physics*, 147(4), 832–852. <http://dx.doi.org/10.1007/s10955-012-0491-0>.
- Benioff, P. (1980). The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by turing machines. *Journal of statistical physics*, 22(5), 563–591.
- Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., & Lloyd, S. (2017). Quantum machine learning. *Nature*, 549(7671), 195–202.
- Bliss, C. A., Frank, M. R., Danforth, C. M., & Dodds, P. S. (2014). An evolutionary algorithm approach to link prediction in dynamic social networks. *Journal of Computer Science*, 5(5), 750–764. <http://dx.doi.org/10.1016/j.jocs.2014.01.003>, URL: <https://www.sciencedirect.com/science/article/pii/S187750314000040>.
- Brassard, G. (1994). Quantum computing: the end of classical cryptography? *ACM SIGACT News*, 25(4), 15–21.
- Cai, H., Zheng, V. W., & Chang, K. C. C. (2018). A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9), 1616–1637.
- Cao, S., Lu, W., & Xu, Q. (2015). GraRep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international conference on information and knowledge management* (pp. 891–900). New York, NY, USA: ACM. <http://dx.doi.org/10.1145/2806416.2806512>, URL: <http://doi.acm.org/10.1145/2806416.2806512>.
- Casteigts, A., Floccini, P., Quattrociochi, W., & Santoro, N. (2012). Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5), 387–408. <http://dx.doi.org/10.1080/17445760.2012.668546>.
- Chakraborty, S., Luh, K., & Roland, J. (2020). How fast do quantum walks mix? *Physical Review Letters*, 124(5), URL: www.scopus.com. Cited By :7.
- Chang, S., Han, W., Tang, J., Qi, G. J., Aggarwal, C. C., & Huang, T. S. (2015). Heterogeneous network embedding via deep architectures. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 119–128).
- Chen, C. S., Chen, S. Y. C., Tsai, A. H. W., & Wei, C. S. (2024). QEEGNet: Quantum machine learning for enhanced electroencephalography encoding. arXiv preprint arXiv:2407.19214.
- Childs, A. M., Cleve, R., Deotto, E., Farhi, E., Gutmann, S., & Spielman, D. A. (2003). Exponential algorithmic speedup by a quantum walk. In *Proceedings of the thirty-fifth annual ACM symposium on theory of computing* (pp. 59–68).
- Chiu, C., & Zhan, J. (2018). Deep learning for link prediction in dynamic networks using weak estimators. *IEEE Access*, 6, 35937–35945. <http://dx.doi.org/10.1109/ACCESS.2018.2845876>.
- Ciliberto, C., Herbster, M., Ialongo, A. D., Pontil, M., Rocchetto, A., Severini, S., et al. (2018). Quantum machine learning: a classical perspective. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2209), Article 20170551.
- Clauset, A., Newman, M. E. J., & Moore, C. (2004). Finding community structure in very large networks. *Physical Review E*, 70, Article 066111. <http://dx.doi.org/10.1103/PhysRevE.70.066111>, URL: <https://link.aps.org/doi/10.1103/PhysRevE.70.066111>.
- Das, S., & Das, S. K. (2017). A probabilistic link prediction model in time-varying social networks. In *2017 IEEE international conference on communications* (pp. 1–6). IEEE.
- David, F., Ryan, G., & Rossi, A. (2012). *A Dynamical System for PageRank with Time-Dependent Teleportation*. Purdue University, (November 20, 2012).
- Dileep, P. R., & Deepthi L., R. (2022). Link prediction in citation networks: A survey. In *2022 third international conference on intelligent computing instrumentation and control technologies* (pp. 1194–1200). <http://dx.doi.org/10.1109/ICICICT54557.2022.9917780>.
- Divakaran, A., & Mohan, A. (2019). Temporal link prediction: a survey. *New Generation Computing*, 1–46.
- Dunjko, V., & Briegel, H. J. (2018). Machine learning & artificial intelligence in the quantum domain: a review of recent progress. *Reports on Progress in Physics*, 81(7), Article 074001.
- Farhi, E., & Gutmann, S. (1998). Quantum computation and decision trees. *Physical Review A - Atomic, Molecular, and Optical Physics*, 58(2), 915–928, URL: www.scopus.com. Cited By :807.
- Farhi, E., & Neven, H. (2018). Classification with quantum neural networks on near term processors. arXiv preprint arXiv:1802.06002.
- Feldman, E., & Hillery, M. (2004). Scattering theory and discrete-time quantum walks. *Physics Letters. A*, 324(4), 277–281. <http://dx.doi.org/10.1016/j.physleta.2004.03.005>, URL: <https://www.sciencedirect.com/science/article/pii/S0375960104003275>.
- Feng, K., Ji, J., Zhang, Y., Ni, Q., Liu, Z., & Beer, M. (2023). Digital twin-driven intelligent assessment of gear surface degradation. *Mechanical Systems and Signal Processing*, 186, Article 109896. <http://dx.doi.org/10.1016/j.ymssp.2022.109896>, URL: <https://www.sciencedirect.com/science/article/pii/S0888327022009645>.
- Feynman, R. P. (2018). Simulating physics with computers. In *Feynman and computation* (pp. 133–153). CRC Press.
- Fire, M., Tenenboim, L., Lesser, O., Puzis, R., Rokach, L., & Elovici, Y. (2011). Link prediction in social networks using computationally efficient topological features. In *2011 IEEE third international conference on privacy, security, risk and trust and 2011 IEEE third international conference on social computing* (pp. 73–80). <http://dx.doi.org/10.1109/PASSAT/SocialCom.2011.20>.
- Fu, W., Song, L., & Xing, E. P. (2009). Dynamic mixed membership blockmodel for evolving networks. In *Proceedings of the 26th annual international conference on machine learning* (pp. 329–336). New York, NY, USA: Association for Computing Machinery, <http://dx.doi.org/10.1145/1553374.1553416>.
- Ganguly, S. (2021). *Quantum machine learning: An applied approach*. Springer.
- Geyer, C. J. (1992). Practical Markov chain Monte Carlo. *Statistical Science*, 7(4), 473–483, URL: <http://www.jstor.org/stable/2246094>.
- Goutte, C., & Gaussier, E. (2005). A probabilistic interpretation of precision, recall and F-score, with implication for evaluation. *3408*, (pp. 345–359). http://dx.doi.org/10.1007/978-3-540-31865-1_25.
- Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on theory of computing* (pp. 212–219). New York, NY, USA: Association for Computing Machinery, <http://dx.doi.org/10.1145/237814.237866>.
- Grover, A., & Leskovec, J. (2016). Node2Vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 855–864). New York, NY, USA: ACM, <http://dx.doi.org/10.1145/2939672.2939754>, URL: <http://doi.acm.org/10.1145/2939672.2939754>.
- Guimerà, R., & Sales-Pardo, M. (2009). Missing and spurious interactions and the reconstruction of complex networks. *Proceedings of the National Academy of Sciences of the United States of America*, 106, 22073–22078. <http://dx.doi.org/10.1073/pnas.0908366106>.
- Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, 30.
- Han, K. H., & Kim, J. H. (2002). Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. *IEEE Transactions on Evolutionary Computation*, 6(6), 580–593. <http://dx.doi.org/10.1109/TEVC.2002.804320>.
- Häner, T., & Steiger, D. S. (2017). 0.5 petabyte simulation of a 45-qubit quantum circuit. In *Proceedings of the international conference for high performance computing, networking, storage and analysis*. ACM, <http://dx.doi.org/10.1145/3126908.3126947>.
- Havlíček, V., Córcoles, A. D., Temme, K., Harrow, A. W., Kandala, A., Chow, J. M., et al. (2019). Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747), 209–212. <http://dx.doi.org/10.1038/s41586-019-0980-2>.
- Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786), 504–507.
- Hirvensalo, M. (2003). Quantum computing. In *Natural computing series*, Springer Berlin Heidelberg, URL: <https://books.google.co.in/books?id=oCd5FWPq7UC>.
- Huang, H. Y., Broughton, M., Mohseni, M., Babbush, R., Boixo, S., Neven, H., et al. (2021). Power of data in quantum machine learning. *Nature Communications*, 12(1), 1–9.
- Huang, Z., Li, X., & Chen, H. (2005). Link prediction approach to collaborative filtering. In *Proceedings of the 5th ACM/IEEE-CS joint conference on digital libraries* (pp. 141–142). <http://dx.doi.org/10.1145/1065385.1065415>, URL: <http://doi.acm.org/10.1145/1065385.1065415>.

- Huang, Z., Silva, A., & Singh, A. (2021). A broader picture of random-walk based graph embedding. (pp. 685–695). New York, NY, USA: Association for Computing Machinery, <http://dx.doi.org/10.1145/3447548.3467300>.
- Huynh, L., Hong, J., Mian, A., Suzuki, H., Wu, Y., & Camtepe, S. (2023). Quantum-inspired machine learning: a survey. *arXiv:2308.11269*.
- Isah, H., Neagu, D., & Trundle, P. (2015). Bipartite network model for inferring hidden ties in crime data. In *Proceedings of the 2015 IEEE/ACM international conference on advances in social networks analysis and mining 2015* (pp. 994–1001). New York, NY, USA: Association for Computing Machinery, <http://dx.doi.org/10.1145/2808797.2808842>.
- Ismael, J. (2021). Quantum Mechanics. In E. N. Zalta (Ed.), *The stanford encyclopedia of philosophy* (Fall 2021). Metaphysics Research Lab, Stanford University.
- Jacot, A., Gabriel, F., & Hongler, C. (2018). *Neural tangent kernel: Convergence and generalization in neural networks* (pp. 8580–8589). Red Hook, NY, USA: Curran Associates Inc.
- Jafarizadeh, M., & Salimi, S. (2007). Investigation of continuous-time quantum walk via spectral distribution associated with adjacency matrix. *Annals of Physics*, 322(5), 1005–1033. <http://dx.doi.org/10.1016/j.aop.2007.01.009>, URL: <https://www.sciencedirect.com/science/article/pii/S0003491607000176>.
- Jiang, Z., Rieffel, E. G., & Wang, Z. (2017). Near-optimal quantum circuit for Grover's unstructured search using a transverse field. *Physical Review A*, 95(6), Article 062317.
- Kadian, K., Garhwal, S., & Kumar, A. (2021). Quantum walk and its application domains: A systematic review. *Computer Science Review*, 41, Article 100419. <http://dx.doi.org/10.1016/j.cosrev.2021.100419>, URL: <https://www.sciencedirect.com/science/article/pii/S1574013721000599>.
- Katok, A., & Thouvenot, J. P. (2006). Chapter 11 - Spectral properties and combinatorial constructions in ergodic theory. In B. Hasselblatt, & A. Katok (Eds.), *Handbook of dynamical systems: vol. 1*, (pp. 649–743). Elsevier Science, [http://dx.doi.org/10.1016/S1874-575X\(06\)80036-6](http://dx.doi.org/10.1016/S1874-575X(06)80036-6), URL: <https://www.sciencedirect.com/science/article/pii/S1874575X06800366>.
- Kerenidis, I., & Prakash, A. (2016). Quantum recommendation systems. *arXiv preprint arXiv:1603.08675*.
- Knill, E., Leibfried, D., Reichle, R., Britton, J., Blakestad, R. B., Jost, J. D., et al. (2008). Randomized benchmarking of quantum gates. *Physical Review A*, 77, Article 012307. <http://dx.doi.org/10.1103/PhysRevA.77.012307>, URL: <https://link.aps.org/doi/10.1103/PhysRevA.77.012307>.
- Konno, N., Namiki, T., & Soshi, T. (2004). Symmetry of distribution for the one-dimensional Hadamard walk. *Interdisciplinary Information Sciences*, 10(1), 11–22. <http://dx.doi.org/10.4036/iis.2004.11>.
- Kumar, M., Mishra, S., & Biswas, B. (2021). Features fusion based link prediction in dynamic networks. *Journal of Computer Science*, Article 101493.
- Kumar, M., Mishra, S., & Biswas, B. (2022a). Features fusion based link prediction in dynamic networks. *Journal of Computer Science*, 57, Article 101493.
- Kumar, M., Mishra, S., & Biswas, B. (2022b). PWA: Path weight aggregation feature for link prediction in dynamic networks. *Computer Communications*, 191, 438–458.
- Kumar, M., Mishra, S., Pandey, R. D., & Biswas, B. (2022). CFLP: A new cost based feature for link prediction in dynamic networks. *Journal of Computer Science*, 62, Article 101726.
- Kumar, A., Singh, S. S., Singh, K., & Biswas, B. (2020). Link prediction techniques, applications, and performance: A survey. *Physica A. Statistical Mechanics and its Applications*, 553, Article 124289. <http://dx.doi.org/10.1016/j.physa.2020.124289>, URL: <https://www.sciencedirect.com/science/article/pii/S0378437120300856>.
- Kumar Sharma, A., Bajpai, B., Adhvaru, R., Dhruvi Pankaj Kumar, S., Parthkumar Gordhanbhai, P., & Kumar, A. (2023). An efficient approach of product recommendation system using nlp technique. *Materials Today: Proceedings*, 80, 3730–3743. <http://dx.doi.org/10.1016/j.matpr.2021.07.371>, URL: <https://www.sciencedirect.com/science/article/pii/S2214785321052305>. SI:5 NANO 2021.
- Kunegis, J. (2017). Reality mining network dataset – KONECT. URL: <http://konect.cc/networks/mit>.
- Leskovec, J., & Krevl, A. (2014). SNAP Datasets: Stanford large network dataset collection. <http://snap.stanford.edu/data>.
- Liu, J., Jiang, Y., Wang, Y., Xie, H., & Ni, J. (2020). Link prediction in dynamic networks based on machine learning. In *2020 3rd international conference on unmanned systems* (pp. 836–841). IEEE.
- Liu, X., Li, X., Fiumara, G., & De Meo, P. (2023). Link prediction approach combined graph neural network with capsule network. *Expert Systems with Applications*, 212, Article 118737. <http://dx.doi.org/10.1016/j.eswa.2022.118737>, URL: <https://www.sciencedirect.com/science/article/pii/S0957417422017559>.
- Liu, M., Liu, J., Liu, R., Makhanov, H., Lykov, D., Apte, A., et al. (2022). Embedding learning in hybrid quantum-classical neural networks. In *2022 IEEE international conference on quantum computing and engineering*. IEEE, <http://dx.doi.org/10.1109/qce53715.2022.00026>.
- Liu, Z., Zhang, Q. M., Lü, L., & Zhou, T. (2011a). Link prediction in complex networks: A local naïve Bayes model. *EPL (Europhysics Letters)*, 96(4), 48007.
- Liu, Z., Zhang, Q. M., Lü, L., & Zhou, T. (2011b). Link prediction in complex networks: A local naïve Bayes model. *EPL (Europhysics Letters)*, 96(4), 48007. <http://dx.doi.org/10.1209/0295-5075/96/48007>.
- Lloyd, S., Mohseni, M., & Rebentrost, P. (2013). Quantum algorithms for supervised and unsupervised machine learning. *arXiv preprint arXiv:1307.0411*.
- Lloyd, S., Mohseni, M., & Rebentrost, P. (2014). Quantum principal component analysis. *Nature Physics*, 10(9), 631–633.
- Ma, X., Sun, P., & Qin, G. (2017). Nonnegative matrix factorization algorithms for link prediction in temporal networks using graph communicability. *Pattern Recognition*, 71, 361–374. <http://dx.doi.org/10.1016/j.patcog.2017.06.025>, URL: <https://www.sciencedirect.com/science/article/pii/S0031320317302480>.
- Magniez, F., Nayak, A., Richter, P. C., & Santha, M. (2012). On the hitting times of quantum versus random walks. *Algorithmica*, <http://dx.doi.org/10.1007/s00453-011-9521-6>.
- Martínez, V., Berzal, F., & Cubero, J.-C. (2016). A survey of link prediction in complex networks. *ACM Computing Surveys (CSUR)*, 49(4), 1–33.
- Mei, Q., Cai, D., Zhang, D., & Zhai, C. (2008). Topic modeling with network regularization. In *Proceedings of the 17th international conference on world wide web* (pp. 101–110).
- Michalski, R., Palus, S., & Kazienko, P. (2011). Matching organizational structure and social network extracted from email communication. In *Lecture notes in business information processing: vol. 87*, (pp. 197–206). Springer Berlin Heidelberg.
- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space, CoRR abs/1301.3781. *arXiv:1301.3781*.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality, CoRR abs/1310.4546. *arXiv:1310.4546*.
- Mitarai, K., Negoro, M., Kitagawa, M., & Fujii, K. (2018). Quantum circuit learning. *Physical Review A*, 98(3), Article 032309.
- Muniz, C. P., Goldschmidt, R., & Choren, R. (2018). Combining contextual, temporal and topological information for unsupervised link prediction in social networks. *Knowledge-Based Systems*, 156, 129–137. <http://dx.doi.org/10.1016/j.knsys.2018.05.027>, URL: <https://www.sciencedirect.com/science/article/pii/S0950705118302569>.
- Narayanan, A., & Moore, M. (1996). Quantum-inspired genetic algorithms. In *Proceedings of IEEE International Conference on Evolutionary Computation* (pp. 61–66).
- Natarajan, N., & Dhillon, I. S. (2014). Inductive matrix completion for predicting gene-disease associations. *Bioinformatics*, 30(12), i60–i68.
- Nguyen, G. H., Lee, J. B., Rossi, R. A., Ahmed, N. K., Koh, E., & Kim, S. (2018). Continuous-time dynamic network embeddings. In *Companion proceedings of the web conference 2018* (pp. 969–976). Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, <http://dx.doi.org/10.1145/3184558.3191526>.
- Ni, Q., Ji, J., Feng, K., Zhang, Y., Lin, D., & Zheng, J. (2024). Data-driven bearing health management using a novel multi-scale fused feature and gated recurrent unit. *Reliability Engineering & System Safety*, 242, Article 109753. <http://dx.doi.org/10.1016/j.res.2023.109753>, URL: <https://www.sciencedirect.com/science/article/pii/S0951832023006671>.
- Orthey, A. C., & Amorim, E. P. (2017). On the spreading of quantum walks starting from local and delocalized states. *arXiv preprint arXiv:1706.06257*.
- Parenti, M. D., & Rastelli, G. (2012). Advances and applications of binding affinity prediction methods in drug discovery. *Biotechnology Advances*, 30(1), 244–250.
- Pati, P., Jaume, G., Foncubierta-Rodriguez, A., Feroce, F., Anniciello, A. M., Scognamiglio, G., et al. (2022). Hierarchical graph representations in digital pathology. *Medical Image Analysis*, 75, Article 102264.
- Peceli, A., Cavalcanti, M. C., & Goldschmidt, R. (2018). Automatic feature selection for supervised learning in link prediction applications: a comparative study. *Knowledge and Information Systems*, 56(1), 85–121.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12, 2825–2830.
- Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). DeepWalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 701–710). New York, NY, USA: ACM, <http://dx.doi.org/10.1145/2623330.2623732>, URL: <http://doi.acm.org/10.1145/2623330.2623732>.
- Perozzi, B., Kulkarni, V., Chen, H., & Skiena, S. (2017a). Don't walk, skip!: Online learning of multi-scale network embeddings. (pp. 258–265). <http://dx.doi.org/10.1145/3110025.3110086>.
- Perozzi, B., Kulkarni, V., Chen, H., & Skiena, S. (2017b). Don't walk, skip! Online learning of multi-scale network embeddings. *arXiv:1605.02115*.
- Portugal, R., Santos, R. A. M., Fernandes, T. D., & Gonçalves, D. N. (2016). The staggered quantum walk model. *Quantum Information Processing*, <http://dx.doi.org/10.1007/s11128-015-1149-z>.
- Preskill, J. (2018). Quantum computing in the NISQ era and beyond. *Quantum*, 2, 79.
- Pudenz, K. L., & Lidar, D. A. (2013). Quantum adiabatic machine learning. *Quantum Information Processing*, 12(5), 2027–2070.
- Qian, K., Eldredge, Z., Ge, W., Pagano, G., Monroe, C., Porto, J. V., et al. (2019). Heisenberg-scaling measurement protocol for analytic functions with quantum sensor networks. *Physical Review A*, 100(4), Article 042304.
- Qu, Z., Liu, X., & Zheng, M. (2022). Temporal-spatial quantum graph convolutional neural network based on Schrödinger approach for traffic congestion prediction. *IEEE Transactions on Intelligent Transportation Systems*, PP, 1–10. <http://dx.doi.org/10.1109/ITITS.2022.3203791>.

- Rebentrost, P., Mohseni, M., & Lloyd, S. (2014). Quantum support vector machine for big data classification. *Physical Review Letters*, 113(13), Article 130503.
- Rehman, M. U. (2024). Quantum-enhanced chaotic image encryption: Strengthening digital data security with 1-D sine-based chaotic maps and quantum coding. *Journal of King Saud University-Computer and Information Sciences*, 36(3), Article 101980.
- Riascos, A. P., & Mateos, J. L. (2021). Random walks on weighted networks: a survey of local and non-local dynamics. *Journal of Complex Networks*, 9(5), cnab032. <http://dx.doi.org/10.1093/comnet/cnab032>.
- Romero, J., Olson, J. P., & Aspuru-Guzik, A. (2017). Quantum autoencoders for efficient compression of quantum data. *Quantum Science and Technology*, 2(4), Article 045001.
- Rossi, R. A., & Ahmed, N. K. (2015). The network data repository with interactive graph analytics and visualization. In AAAI. URL: <https://networkrepository.com>.
- Rozemberczki, B., Kiss, O., & Sarkar, R. (2020). Karate Club: An API Oriented Open-source Python Framework for Unsupervised Learning on Graphs. In *Proceedings of the 29th ACM international conference on information and knowledge management* (pp. 3125–3132). ACM.
- Sarkar, P., Chakrabarti, D., & Jordan, M. (2012). Nonparametric link prediction in dynamic networks. arXiv preprint arXiv:1206.6394.
- Sasaki, M., & Carlini, A. (2002). Quantum learning and universal quantum matching machine. *Physical Review A*, 66(2), Article 022303.
- Schuld, M. (2021). Quantum machine learning models are kernel methods. arXiv e-prints, arXiv-2101.
- Shaw, B., & Jebara, T. (2009). Structure preserving embedding. In *Proceedings of the 26th annual international conference on machine learning* (pp. 937–944).
- Shor, P. (1994). Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science* (pp. 124–134). <http://dx.doi.org/10.1109/SFCS.1994.365700>.
- Singh, N., Kumar, M., & Biswas, B. (2024). PQCLP: Parameterized quantum circuit based link prediction in dynamic networks. *Computer Networks*, 241, Article 110210.
- Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015). Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web* (pp. 1067–1077).
- Terashi, K., Kaneda, M., Kishimoto, T., Saito, M., Sawada, R., & Tanaka, J. (2021). Event classification with quantum machine learning in high-energy physics. *Computing and Software for Big Science*.
- Van Gael, J. (2005). The role of interference and entanglement in quantum computing. *Semantic Scholar*.
- Vital, A., Silva, F., & Amancio, D. (2024). Comparing random walks in graph embedding and link prediction. *PLOS ONE*, 19, <http://dx.doi.org/10.1371/journal.pone.0312863>.
- Wan, K. H., Dahlsten, O., Kristjánsson, H., Gardner, R., & Kim, M. (2017). Quantum generalisation of feedforward neural networks. *npj Quantum information*, 3(1), 1–8.
- Wang, X., Cui, P., Wang, J., Pei, J., Zhu, W., & Yang, S. (2017). Community preserving network embedding. In *Thirty-first AAAI conference on artificial intelligence*.
- Wang, S., Tang, J., Morstatter, F., & Liu, H. (2016). Paired restricted boltzmann machine for linked data. In *Proceedings of the 25th ACM international conference on information and knowledge management* (pp. 1753–1762).
- Wang, P., Xu, B., Wu, Y., & Zhou, X. (2015). Link prediction in social networks: the state-of-the-art. *Science China. Information Sciences*, 58(1), 1–38.
- Whitfield, J. D., Rodríguez-Rosario, C. A., & Aspuru-Guzik, A. (2010). Quantum stochastic walks: A generalization of classical random walks and quantum walks. *Physical Review A*, 81, Article 022323. <http://dx.doi.org/10.1103/PhysRevA.81.022323>, URL: <https://link.aps.org/doi/10.1103/PhysRevA.81.022323>.
- Wong, T. G. (2017). Coined quantum walks on weighted graphs. *Journal of Physics. A. Mathematical and Theoretical*, 50(47), Article 475301. <http://dx.doi.org/10.1088/1751-8121/aa8c17>.
- Xu, M., Jin, R., & Zhou, Z.-H. (2013). Speedup matrix completion with side information: Application to multi-label learning. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K. Weinberger (Eds.), *Advances in neural information processing systems*: vol. 26, Curran Associates, Inc., URL: https://proceedings.neurips.cc/paper_files/paper/2013/file/e58cc5ca94270acaceed13bc82dfedf7-Paper.pdf.
- Yan, B., Wang, G., Yu, J., Jin, X., & Zhang, H. (2021). Spatial-temporal chebyshev graph neural network for traffic flow prediction in iot-based its. *IEEE Internet of Things Journal*, 9(12), 9266–9279.
- Yan, S., Xu, D., Zhang, B., Zhang, H.-J., Yang, Q., & Lin, S. (2006). Graph embedding and extensions: A general framework for dimensionality reduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(1), 40–51.
- Yang, C., Liu, Z., Zhao, D., Sun, M., & Chang, E. Y. (2015). Network representation learning with rich text information. In *International joint conference on artificial intelligence*. URL: <https://api.semanticscholar.org/CorpusID:2452205>.
- Yano, H., Suzuki, Y., Itoh, K., Raymond, R., & Yamamoto, N. (2021). Efficient discrete feature encoding for variational quantum classifier. *IEEE Transactions on Quantum Engineering*, 2, 1–14. <http://dx.doi.org/10.1109/tqe.2021.3103050>.
- Yasami, Y., & Safaei, F. (2018). A novel multilayer model for missing link prediction and future link forecasting in dynamic complex networks. *Physica A. Statistical Mechanics and its Applications*, 492(C), 2166–2197. <http://dx.doi.org/10.1016/j.physa.2017.11.1>, URL: <https://ideas.repec.org/a/eee/phsmap/v492y2018icp2166-2197.html>.
- Zhang, D., Yin, J., Zhu, X., & Zhang, C. (2016). Homophily, structure, and content augmented network representation learning. In *2016 IEEE 16th international conference on data mining* (pp. 609–618). IEEE.
- Zhiwei, C., Yichao, Z., Jihong, G., & Shuigeng, Z. (2018). Link prediction based on quantum-inspired ant colony optimization. *Scientific Reports*, <http://dx.doi.org/10.1038/s41598-018-31254-3>.
- Zhou, M., Kong, Y., Zhang, S., Liu, D., & Jin, H. (2020). The deep fusion of topological structure and attribute information for link prediction. *IEEE Access*, 8, 34398–34406.
- Zhu, L., Guo, D., Yin, J., Steeg, G., & Galstyan, A. (2016). Scalable temporal latent space inference for link prediction in dynamic social networks. *IEEE Transactions on Knowledge and Data Engineering*, 28(10), 2765–2777. <http://dx.doi.org/10.1109/TKDE.2016.2591009>, Publisher Copyright: © 1989-2012 IEEE.
- Zhu, Y., Ye, Z., Zhao, H., & Zhang, K. (2018). Network representation learning based on community and text features. In *China national conference on Chinese computational linguistics*. URL: <https://api.semanticscholar.org/CorpusID:52974608>.



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



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



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