GraphQNTK: Quantum Neural Tangent Kernel for Graph Data

Yehui Tang, Junchi Yan[*]
Department of Computer Science and Engineering
MoE Key Lab of Artificial Intelligence
Shanghai Jiao Tong University
{yehuitang, yanjunchi}@sjtu.edu.cn

Abstract

Graph Neural Networks (GNNs) and Graph Kernels (GKs) are two fundamental tools used to analyze graph-structured data. Efforts have been recently made in developing a composite graph learning architecture combining the expressive power of GNNs and the transparent trainability of GKs. However, learning efficiency on these models should be carefully considered as the huge computation overhead. Besides, their convolutional methods are often straightforward and introduce severe loss of graph structure information. In this paper, we design a novel quantum graph learning model to characterize the structural information while using quantum parallelism to improve computing efficiency. Specifically, a quantum algorithm is proposed to approximately estimate the neural tangent kernel of the underlying graph neural network where a multi-head quantum attention mechanism is introduced to properly incorporate semantic similarity information of nodes into the model. We empirically show that our method achieves competitive performance on several graph classification benchmarks, and theoretical analysis is provided to demonstrate the superiority of our quantum algorithm. Source code is available at `https://github.com/abel1231/graphQNTK`.

1 Introduction

Fusing quantum computing and classic machine learning has become a promising subject of research. Quantum-based algorithms have been proposed in recent years, from naive quantum non-parametric machine learning [52, 36, 43, 32] to classic-quantum hybrid deep leaning [7, 10, 46, 37, 14]. Despite that quantum machine learning (QML) has shown its potential in many machine learning tasks, quantum computing for graph learning is still in its early stage [61]. Inspired by the two popular classes of methods for learning on graph data, i.e., Graph Neural Networks (GNNs) [20, 39, 21, 67] and Graph Kernels (GKs) [23], several works attempt to build quantum graph learning architecture that captures the structural information of graph data, such as Quantum Graph Neural Networks (QGNNs) [64, 7, 11, 16, 1] and Quantum Graph Kernel Methods (QGKs) [56, 3, 25, 4]. A brief review about quantum graph learning is illustrated in Fig. 1.

Some quantum subroutines for attribute encoding [5, 70] and structural encoding [64, 46] have been developed to dissolve the characteristics of the graph into the quantum model. However, most present quantum graph learning models are hybrid such that the expressive capability depends more on the complexity of the classic modules [70]. It is difficult to characterize the structure information and attribute information of the graph by the quantum components without the participation of classic modules. Even worse, the frequent interactions between classical systems and quantum environments generally incur additional overhead [55]. It is unclear whether the introduced quantum module

[*] Junchi Yan is the correspondence author who is also with Shanghai AI Laboratory.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

can improve the performance of the model as well as the training efficiency. Besides, most of existing proposals for quantum machine learning for graphs lack a clear demonstration of a quantum superiority for tasks on classical datasets.

Using quantum computing power to boost the trainability and expressive behaviour of classic machine learning models provides one of the most promising direction for quantum machine learning. It is demonstrated that the power of quantum computing could be used to find atypical but useful patterns that classical systems are not considered to be able to generate effectively [14, 24, 28], and accelerate the training process of existing classic models [36, 43, 71]. Several quantum algorithms [52, 45, 44] based on the HHL algorithm [22] show the exponential speedup compared with their classical counterpart, with a assumption that a quantum random access memory (QRAM) [35] is accessible. Recent literature employ quantum algorithms to efficiently train deep neural networks [71], reconstruct unsupervised clustering [34] and supervised kernel classifier [43]. It is hopeful that quantum computing could provide a new learning paradigm. In addition, simulations and physical experiments have proved the potential of using quantum algorithms to encode and process regular classical data such as text and image [60, 6].

Beyond vanilla GNNs and GKs, composite graph learning studies have emerged that combine the advantages of both areas [50, 10, 18]. In this literature, Graph Neural Tangent Kernel (GNTK) [17] based on neural tangent kernel (NTK) [31] shines its lights on elegantly fusing GNNs and GKs, leading to new prospectives of training and analysing infinite-width GNNs. However, the computation overheads is extremely large due to either the dense gram matrix [17], or the large number of substructures to be compared after graph decomposition [10]. Prospectively, the barrier that conventional model is difficult to train and scale up is expected to be circumvented with the help of the uniqueness of quantum computing. Early research involves altering the amplitude of quantum basis states to accomplish a quantum logic operations [8], which is profitable from the huge quantum Hilbert space to encode the normalized data. Recently, simultaneous transformation of basic states in quantum superposition using quantum parallelism is regarded as a remarkable manifestation of quantum superiority, which is successfully implemented in classic machine learning to reduce the computational overheads [36, 37, 71]. These strategies could be helpful in the regime of training graph models with either the non-convex nature of the training procedure, or the poor scalability w.r.t. training size.

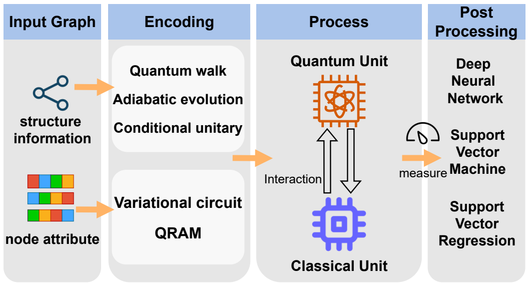

Figure 1: Overview of quantum graph learning.

In this paper, we focus on quantum machine learning of graph-structured data with attributed nodes and binary edges. Inspired by recent quantum neural network methods [37, 71] that efficiently reconstruct the dynamics of classic neural networks using quantum computing techniques, a new quantum graph learning model is proposed which is analogue to train an infinite-width GNN with attention mechanism, where the number of heads goes to infinity. Our contributions are:

- **Attention-enhanced GNTK.** Infinite-width GNN is a well established term in the GNN literature [17, 29], and GNTK [17] is a powerful tool to analyze the GNN. However, the traditional feature aggregation of the vanilla GNTK is straightforward, limiting itself within joint neighborhoods. In this paper, a multi-head attention mechanism is introduced to properly incorporate semantic similarity information of disconnected nodes but with similar features to improve model expressiveness.
- **Kernel methods for evaluating the dynamics of wide and deep GNNs.** It is generally hard to train a deep GNN with attention, especially when the width of GNN, the width of attention layer, or the number of heads goes to large. We properly incorporate infinite-width multi-head attention into GNTK by using NTK theory. We use kernel methods to capture the dynamics of infinite-width GNN with infinite-width attention, thus avoiding the huge overhead of training a wide and deep GNN.
- **Speedup introduced by quantum computing.** Although GNTK is a useful method to train an infinite-width GNN, its cost still grows quadratically with respect to the volume of data, which is intractable for large datasets. We re-design the Attention-enhanced GNTK by splitting it into small components and reuniting them using the quantum linear algebra subroutines. The produced

quantum graph kernel – GraphQNTK, theoretically reduces the computational complexity from $O(N^2)$ to $O(N)$ benefited from the quantum parallelism.

2 Methodology

2.1 Preliminaries

We first briefly review the most common setting for GNNs and the corresponding NTK, and by the way the notation is given. A graph $G = (V, E)$ is denoted by a collection of nodes V and edges E. Each node has a d-dimensional feature vector $\mathbf{h}_v \in \mathbb{R}^d$, $v \in V$, and $\mathbf{H} \in \mathbb{R}^{n \times d}$ is the feature matrix stacking all nodes features. For graph classification, we consider the dataset with a set of graphs $\{G_1, \ldots, G_N\} \subseteq \mathcal{G}$ and their labels $\{y_1, \ldots, y_N\} \subseteq \mathcal{Y}$. Our goal is to learn to predict labels of unseen graphs.

The formulation of GNN. The differences of GNNs mainly depend on the different settings of message propagation process. Here we consider a simple message passing framework [20] and the propagation of the l-th ($l \in [L]$) layer is given as:

$$\hat{\mathbf{h}}_u^l = \sum_{v \in \mathcal{N}(u) \cup \{u\}} \mathbf{h}_v^{(l-1)}, \tag{1}$$

$$\mathbf{h}_u^l = \sqrt{\frac{c_\sigma}{d_R^l}} \sigma \left(\mathbf{W}_R^l \sqrt{\frac{c_\sigma}{d_{R-1}^l}} \sigma \left(\mathbf{W}_{R-1}^l \cdots \sqrt{\frac{c_\sigma}{d_1^l}} \cdot \sigma \left(\mathbf{W}_1^l \hat{\mathbf{h}}_u^l \right) \right) \right), \tag{2}$$

where $\mathcal{N}(u)$ denotes the neighbors of u, c_σ is the scaling factor, d_r^l is the output dimension of the l-th layer and the r-th fully-connected layers, σ is an element-wise activated function, and $\mathbf{W}_R^l$ is learnable weights performing on the input for R times of the l-th layer (equivalent to R fully-connected layers without the bias term).

For graph classification, the output is a permutation invariance function acting on the collection of all node features in the last layer. The popular `sum_pooling` function is adopted: $\mathbf{h}_G = \sum_{u \in V} \mathbf{h}_u^L$..

NTK of the infinite-width GNN. Consider a training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^N \subset \mathbb{R}^d \times \mathbb{R}$. When an over-parameterized fully connected network $f(\theta, \mathbf{x}) : \mathbb{R}^d \to \mathbb{R}$ whose width is allowed to go to infinity and parameters θ are randomly initialized and trained with gradient descent, the dynamics of the network is equivalent to the kernel regression [31]. This is the so called neural tangent kernel (NTK):

$$\mathbf{H}(t)_{ij} = \left\langle \frac{\partial f\left(\theta(t), x_i\right)}{\partial \theta}, \frac{\partial f\left(\theta(t), x_j\right)}{\partial \theta} \right\rangle, \tag{3}$$

which remains constant during training, i.e., $\mathbf{H}(t) = \mathbf{H}(0)$. And we replace $\mathbf{H}(t)$ with $\mathbf{H}$ for convenience. The final prediction for a test datapoint $\mathbf{x}_*$ is

$$f(\mathbf{x}_*) = \mathbf{k}_* \mathbf{H}^{-1} \mathbf{y}, \tag{4}$$

where $\mathbf{y}_i = y_i$ and $\mathbf{k}_* \in \mathbb{R}^N$ is the vector whose i-th element denotes the NTK value between $\mathbf{x}_i$ and $\mathbf{x}_*$.

It is discovered that convolutional neural networks (CNNs) with infinite-width channels and infinite number of filters also have the same behaviour [2]. Inspired by this, Du et al. [17] adopts the designing strategy of NTK and leverages a GNN architecture to design new graph kernels, which is called GNTK. The dynamics of training the GNTK is equivalent to train an infinitely-wide GNN initialized with random weights trained with gradient descent. Specifically, consider two input graph $G = (V, E)$ and $G' = (V', E')$ with $|V| = n$ and $|V| = n'$, the GNTK $\mathbf{\Theta} \in \mathbb{R}^{n \times n'}$ and the relative covariance matrix $\mathbf{\Sigma} \in \mathbb{R}^{n \times n'}$ in the l-th layer of the feature aggregation phase as described in Eq. 1 after R fully-connected layers are given by

$$\begin{aligned} \left[\mathbf{\Sigma}_0^l \left(G, G'\right)\right]_{uu'} &= \sum_{v \in \mathcal{N}(u) \cup \{u\}} \sum_{v' \in \mathcal{N}(u') \cup \{u'\}} \left[\mathbf{\Sigma}_R^{l-1} \left(G, G'\right)\right]_{vv'}, \\ \left[\mathbf{\Theta}_0^l \left(G, G'\right)\right]_{uu'} &= \sum_{v \in \mathcal{N}(u) \cup \{u\}} \sum_{v' \in \mathcal{N}(u') \cup \{u'\}} \left[\mathbf{\Theta}_R^{l-1} \left(G, G'\right)\right]_{vv'}, \end{aligned} \tag{5}$$

which is an affine transformation of the input GNTK and covariance respectively where $\left[\mathbf{\Theta}_R^0(G,G')\right]_{uu'}$ and $\left[\mathbf{\Sigma}_R^0(G,G')\right]_{uu'}$ are both defined to be $\mathbf{h}_u^\top \mathbf{h}_{u'}$. We replace them with $\left[\mathbf{\Theta}^0(G,G')\right]_{uu'}$ and $\left[\mathbf{\Sigma}^0(G,G')\right]_{uu'}$ respectively without ambiguity.

The successive fully-connected layers defined in Eq. 2 are used to update the node hidden feature after aggregation. Specifically, the GNTK of the fully-connected layer is recursively associated to that of the previous layer, and the transformation is given by

$$\left[\mathbf{\Sigma}_r^l\left(G,G'\right)\right]_{uu'} = \hat{\sigma}^{(r-1)}\left(\left[\mathbf{\Sigma}_{r-1}^l\left(G,G'\right)\right]_{uu'}\right), r \in [R], \tag{6}$$

where $\hat{\sigma}^{(r)} : [-1,1] \rightarrow \mathbb{R}$ denotes the the conjugate activation function corresponding to the activated function σ with centered Gaussian processes of covariance at the r-th fully-connected layer, as described in [15]. And the derivation of the covariance is

$$\left[\dot{\mathbf{\Sigma}}_r^l\left(G,G'\right)\right]_{uu'} = \hat{\dot{\sigma}}\left(\hat{\sigma}^{(r-1)}\left(\left[\mathbf{\Sigma}_{r-1}^l\left(G,G'\right)\right]_{uu'}\right)\right), \tag{7}$$

where $\hat{\dot{\sigma}}$ denotes the derivative of σ. Given Eq. 6 and Eq. 7, the transformation of the GNTK for the feature update phase denoted by Eq. 2 is given by

$$\left[\mathbf{\Theta}_R^l\left(G,G'\right)\right]_{uu'} = \sum_{r=1}^{R}\left[\mathbf{\Sigma}_0^l\left(G,G'\right)\right]_{uu'}\left(\prod_{r'=r}^{R}\left[\dot{\mathbf{\Sigma}}_0^l\left(G,G'\right)\right]_{uu'}\right). \tag{8}$$

Therefore, computing each element of the GNTK (or covariance) matrix is only reliant on the element at the same place of the GNTK (or covariance) matrix in the previous fully-connected layer. The final GNTK corresponding the two input graphs G and G' determined by the `sum_pooling` function:

$$\mathbf{\Theta}\left(G,G'\right) = \sum_{u\in V, u'\in V'}\left[\mathbf{\Theta}_R^L\left(G,G'\right)\right]_{uu'}. \tag{9}$$

Intuitively, calculating each element of the GNTK of fully-connected layers could be accelerating by a proper quantum kernel estimation algorithm. However, it is indirect to realize an end-to-end speedup for GNTK since calculating the element of GNTK requires an affine transformation. To circumvent this barrier, we derive a unitary quantum aggregation transformation to bridge the gap between quantum kernel methods and estimation of GNTK.

2.2 QNTK with Attention Mechanism

Before giving the analytical quantum reconstruction of GNTK with multi-head attention mechanism, we first elaborate on how to integrate the transformer layer into the GNN as described in Sec. 2.1. The resulting GraphQNTK can be efficiently reconstructed by quantum computing paradigm, which gives a quadratic speed-up over the classic estimation of GNTK. The mechanism to build the GNN and estimate the GNTK is shown in Fig. 2.

GNN with multi-head attention. The aggregation process of vanilla GCN [39] regards the contribution of each node's neighbor to the central node as equally important, which can be viewed as learning an averaged filter across the whole graph [66], leading to a great loss of structure information. Besides, the aggregation only is performed within the adjoining neighbors under the assumption that the graph is homophilous. The method may fail to learn effective graph structures for message passing [12]. To capture the global node similarity semantics of the provided graph, numerous attempts that employ transformer for graph learning have been developed [27, 51, 53, 68]. Consider the input feature matrix $\mathbf{H}_{\text{in}}^l \in \mathbb{R}^{N\times s^l}$ where N denotes the number of samples and s^l is the dimension of feature at layer l before implementation of the transformer. The single transformer layer is to project the input $\mathbf{H}_{\text{in}}^l \in \mathbb{R}^{N\times s^l}$ by three matrices, i.e., $\mathbf{W}_Q^l \in \mathbb{R}^{s^l\times s_K^l}$, $\mathbf{W}_K^l \in \mathbb{R}^{s^l\times s_K^l}$ and $\mathbf{W}_V^l \in \mathbb{R}^{s^l\times s_V^l}$, to the corresponding representations $\mathbf{Q}^l, \mathbf{K}^l, \mathbf{V}^l$. The formulation is given as

$$\begin{gathered}\mathbf{Q}^l = \mathbf{H}_{\text{in}}^l\mathbf{W}_Q^l, \quad \mathbf{K}^l = \mathbf{H}_{\text{in}}^l\mathbf{W}_K^l, \quad \mathbf{V}^l = \mathbf{H}_{\text{in}}^l\mathbf{W}_V^l, \\ \widehat{\mathbf{H}}^l = \zeta\left(\mathbf{G}^l\right)\mathbf{V}^l, \quad \mathbf{G}^l = \frac{\mathbf{Q}^l{\mathbf{K}^l}^\top}{\sqrt{s_K^l}},\end{gathered} \tag{10}$$

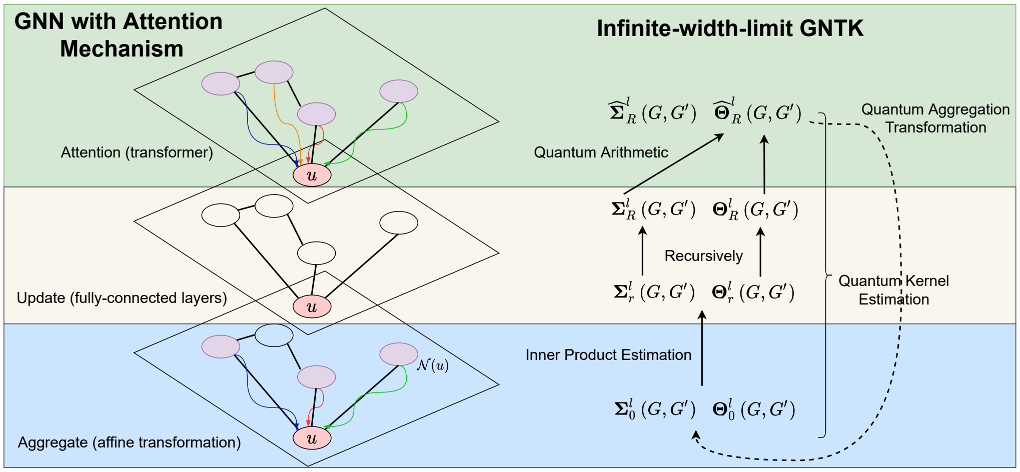
GNN with Attention Mechanism
Infinite-width-limit GNTK
$\widehat{\boldsymbol{\Sigma}}^{l}_{R}\left(G, G'\right)$
$\widehat{\boldsymbol{\Theta}}^{l}_{R}\left(G, G'\right)$
Quantum Aggregation Transformation
Attention (transformer)
Quantum Arithmetic
u
$\boldsymbol{\Sigma}^{l}_{R}\left(G, G'\right)$
$\boldsymbol{\Theta}^{l}_{R}\left(G, G'\right)$
Recursively
Update (fully-connected layers)
u
$\boldsymbol{\Sigma}^{l}_{r}\left(G, G'\right)$
$\boldsymbol{\Theta}^{l}_{r}\left(G, G'\right)$
Quantum Kernel Estimation
Inner Product Estimation
$\mathcal{N}(u)$
$\boldsymbol{\Sigma}^{l}_{0}\left(G, G'\right)$
$\boldsymbol{\Theta}^{l}_{0}\left(G, G'\right)$
Aggregate (affine transformation)
u

Figure 2: Framework for GNN with attention mechanism and its corresponding GNTK. The GNN comprises a message transmission process similar to the vanilla GCN but involves a transformer at the tail of the model (excluding the last layer), which characterizes the global semantic similarity between each pairs of nodes. The neighbor aggregation is kept since the two nodes connected by an edge often have stronger semantic relationship. The dynamics of the infinite-width-limit GNN is analogous to kernel methods and we reconstruct it by quantum algorithms to estimate the kernel.

where ζ denotes an element-wise activated function. The multi-head attention alternative is given by

$$\mathbf{H}^{l}_{\text{out}} = \left[\widehat{\mathbf{H}}^{l}_{\text{head}_1}, \ldots, \widehat{\mathbf{H}}^{l}_{\text{head}_M}\right] \mathbf{W}^{l}_{O}, \tag{11}$$

where $\mathbf{W}^{l}_{O} \in \mathbb{R}^{(M s^{l}_{V}) \times s^{l}}$ projects the $N \times M s^{l}_{V}$ concatenated multi-head feature matrix back to $N \times s^{l}$ matrix.

Let $\mathbf{Y}$ and $\boldsymbol{\Theta}$ denote the neural network Gaussian Process Kernel (NNGP) [51] and NTK after the transformer layer, and let $\widetilde{\mathbf{Y}}$ and $\widetilde{\boldsymbol{\Theta}}$ be the input NNGP and NTK before the transformer layer. Consider two input feature vector $\mathbf{x}$ and $\mathbf{x}'$. When the output dimension of the transformer layer and the number of heads go to infinity, i.e., $s^{l} \to \infty, s^{l}_{K} \to \infty, s^{l}_{V} \to \infty, M \to \infty$, the output NTK is:

$$\begin{aligned} \boldsymbol{\Theta}\left(\mathbf{x}, \mathbf{x}'\right) &= 2\mathbf{Y}\left(\mathbf{x}, \mathbf{x}'\right) + \zeta\left(\widetilde{\mathbf{Y}}\left(\mathbf{x}, \mathbf{x}\right)\right) \widetilde{\boldsymbol{\Theta}}\left(\mathbf{x}, \mathbf{x}'\right) \zeta\left(\widetilde{\mathbf{Y}}\left(\mathbf{x}', \mathbf{x}'\right)\right)^{\top}, \\ \mathbf{Y}\left(\mathbf{x}, \mathbf{x}'\right) &= \zeta\left(\widetilde{\mathbf{Y}}\left(\mathbf{x}, \mathbf{x}\right)\right) \widetilde{\mathbf{Y}}\left(\mathbf{x}, \mathbf{x}'\right) \zeta\left(\widetilde{\mathbf{Y}}\left(\mathbf{x}', \mathbf{x}'\right)\right)^{\top}, \end{aligned} \tag{12}$$

where the under the restriction that 1) $\mathbf{W}^{l}_{Q}$ and $\mathbf{W}^{l}_{K}$ share the same weighs, and 2) scaling the dot products between $\mathbf{Q}^{l}$ and $\mathbf{K}^{l}$ by their dimension instead of the square root of the same quantity, i.e., $\mathbf{G}^{l} = \frac{\mathbf{Q}^{l} \mathbf{K}^{l\top}}{s^{l}_{K}}$. The detailed proof can be found in [26].

To efficiently estimate the element of the NTK defined by the transformer layer using quantum parallel, we slightly modify the Eq. 12 as

$$\begin{aligned} \boldsymbol{\Theta}\left(\mathbf{x}, \mathbf{x}'\right) &= 2\mathbf{Y}\left(\mathbf{x}, \mathbf{x}'\right) + \widetilde{\mathbf{T}}_{\zeta}\left(\mathbf{x}, \mathbf{x}'\right) \odot \widetilde{\boldsymbol{\Theta}}\left(\mathbf{x}, \mathbf{x}'\right), \\ \mathbf{Y}\left(\mathbf{x}, \mathbf{x}'\right) &= \widetilde{\mathbf{T}}_{\zeta}\left(\mathbf{x}, \mathbf{x}'\right) \odot \widetilde{\mathbf{Y}}\left(\mathbf{x}, \mathbf{x}'\right), \end{aligned} \tag{13}$$

where $\widetilde{\mathbf{T}}\left(\mathbf{x}, \mathbf{x}'\right)$ is the result of matrix multiplication between the column vector of the diagonal of $\widetilde{\mathbf{Y}}\left(\mathbf{x}, \mathbf{x}\right)$ and row vector of the diagonal of $\widetilde{\mathbf{Y}}\left(\mathbf{x}', \mathbf{x}'\right)$, and $\widetilde{\mathbf{T}}_{\zeta}$ is the result of matrix multiplication between the diagonal of those two matrix after activated operation. For simplicity of use, we consider identity function as the activated operation, i.e, $\zeta = I$. It is reasonable to accept this modification since in the limit of infinite width neural network the output converges in distribution to a multivariate normal with a block diagonal covariance [51]. Notice that the difference between the definition of NNGP and the covariance of NTK is that the former denotes the expectation with respect to the output before the activated operation, while the later after the denotes the expectation with respect to the output after the activated operation [31]. Consequently, we consider that $\mathbf{Y}$ is equal to the covariance of NTK within the transformer layer as the result of the identity activated function.

GNTK with infinite-width-limit attention. To appropriately incorporate semantic similarity information of nodes into the model, a multi-head attention mechanism is implemented at the tail of the each GNN layer except the first and the last layer, and the calculation of the GNTK with infinite-width-limit attention is to insert an additional procedure after the fully connected layers. For the two input graphs G and G', the formulation derived by Eq. 13 is given as

$$\begin{aligned}\left[\widehat{\mathbf{\Theta}}_R^l\left(G,G'\right)\right]_{uu'} &= 2\left[\widehat{\mathbf{\Sigma}}_R^l\left(G,G'\right)\right]_{uu'} + \left[\mathbf{T}^l\left(G,G'\right)\right]_{uu'}\left[\mathbf{\Theta}_R^l\left(G,G'\right)\right]_{uu'}, \\ \left[\widehat{\mathbf{\Sigma}}_R^l\left(G,G'\right)\right]_{uu'} &= \left[\mathbf{T}^l\left(G,G'\right)\right]_{uu'}\left[\mathbf{\Sigma}_R^l\left(G,G'\right)\right]_{uu'},\end{aligned} \tag{14}$$

where $\mathbf{T}^l\left(G,G'\right)$ is the result of matrix multiplication between the column vector of the diagonal of $\mathbf{\Sigma}_R^l\left(G,G\right)$ and row vector of the diagonal of $\mathbf{\Sigma}_R^l\left(G',G'\right)$. The affine transformation of the input GNTK corresponding to the aggregation phase as described in Eq. 5 is changed to (similar to $\mathbf{\Sigma}_0^l$):

$$\left[\mathbf{\Theta}_0^l\left(G,G'\right)\right]_{uu'} = \sum_{v\in\mathcal{N}(u)\cup\{u\}}\sum_{v'\in\mathcal{N}(u')\cup\{u'\}}\left[\widehat{\mathbf{\Theta}}_R^{l-1}\left(G,G'\right)\right]_{vv'}, \tag{15}$$

2.3 The Proposed GraphQNTK

We first show that estimating the single-layer GraphQNTK and its covariance with infinite-width-limit attention mechanism can be efficiently reconstructed in the regime of quantum computing, and generalize to the multi-layer model. The following statements only consider two input graphs $G = (V,E)$ and $G' = (V',E')$ with $|V| = n$ and $|V'| = n'$, and the corresponding feature matrix $\mathbf{H} = [\mathbf{h}_1^\top,\cdots,\mathbf{h}_u^\top,\cdots,\mathbf{h}_n^\top] \in \mathbb{R}^{n\times d}$ and $\mathbf{H}' = [\mathbf{h}_1^\top,\cdots,\mathbf{h}_{u'}^\top,\cdots,\mathbf{h}_{n'}^\top] \in \mathbb{R}^{n'\times d}$. The approximate estimation of GNTK is denoted as $\bar{\mathbf{\Theta}} \in \mathbb{R}^{n\times n'}$ and its element is $\bar{\mathbf{\Theta}}_{uu'}$. The corresponding covariance is $\bar{\mathbf{\Sigma}} \in \mathbb{R}^{n\times n'}$ and $\bar{\mathbf{\Sigma}}_{uu'}$. We use $\bar{\mathbf{\Theta}}_{GG'} \in \mathbb{R}$ to represent GraphQNTK after readout. We omit the subscript R for clarity. The same setting can be easily generalized to the arbitrary pair of graphs $G, G' \in \mathcal{G}$ by introducing auxiliary index registers. First, we introduce the quantum data structure accessible to the classical data, as commonly used by QML algorithms [52, 36, 34, 71].

Feature encoding. Using the storage structure as stated in our proposed Theorem 1 in Appendix, the feature matrix can be prepared into the QRAM at the initialization of the algorithm. The data encoding only occurs a single time and readout operation only takes logarithmic complexity time with respect to the number of samples n and dimension of feature d. The quantum representations corresponding to the encoded feature vector and feature matrix are as follows

$$\begin{aligned}&|u\rangle\,|0\rangle \to |u\rangle\,|\mathbf{h}_u\rangle\,, \quad |0\rangle \to \frac{1}{\|\mathbf{H}\|_F}\sum_u \|\mathbf{h}_u\|\,|u\rangle, \\ &|u'\rangle\,|0\rangle \to |u'\rangle\,|\mathbf{h}_{u'}\rangle\,, \quad |0\rangle \to \frac{1}{\|\mathbf{H}'\|_F}\sum_{u'} \|\mathbf{h}_{u'}\|\,|u'\rangle.\end{aligned} \tag{16}$$

Estimation of the initialized NTK. The empirical uncentered covariance of inputs $\left[\mathbf{\Sigma}^0(G,G')\right]_{uu'}$ and the initialized GNTK $\left[\mathbf{\Theta}^0(G,G')\right]_{uu'}$ is the inner product between $\mathbf{h}_u$ and $\mathbf{h}_{u'}$. Following a similar approach to [37], the inner product between two vectors with respect to their quantum representations can be estimate efficiently by introducing an auxiliary register. Specifically, estimation of the inner product $\mathbf{h}_u^\top\mathbf{h}_{u'}$ can be performed by constructing the state $\frac{1}{\sqrt{2}}(|u\rangle|u'\rangle|0\rangle|\,|\mathbf{h}_u\rangle\rangle + |u\rangle|u'\rangle|1\rangle|\,|\mathbf{h}_{u'}\rangle\rangle)$. Applying a Hadamard gate on the third register gives the state $|u\rangle|u'\rangle\left(\sqrt{P_{uu'}}\,|0,g_{uu'}\rangle + \sqrt{1-P_{uu'}}\,|1,g'_{uu'}\rangle\right)$, where $P_{uu'} = \frac{1+\mathbf{h}_u^\top\mathbf{h}_{u'}}{2}$ is the estimation of the inner product. This procedure takes $O(\log d)$ time and we denote this quantum operation by $\mathcal{D}^0$, and we add a subscript to denote the corresponding conditioned operator, i.e, $\mathcal{D}_{uu'}^0$ represents $\mathcal{D}^0$ is conditioned acting on the basis state coupled with state $|0\rangle \to |u\rangle|u'\rangle$. We can perform the $\mathcal{D}_{uu'}^0$ in superposition such that the state $\frac{1}{\sqrt{nn'}}\sum_{u\in V}\sum_{u'\in V'}|u\rangle|u'\rangle\left(\sqrt{P_{uu'}}\,|0,g_{uu'}\rangle + \sqrt{1-P_{uu'}}\,|1,g'_{uu'}\rangle\right)$ can be generated in time $O(\log(nd))$.

Quantum aggregation transformation. Recall that an affine transformation (refer to Eq. 5 and Eq. 15) acting on the GNTK and its covariance is relative to the neighborhood aggregation defined by Eq. 1. Therefore, it is indirect to realize an end-to-end speedup similar to the estimation of the inner product since the transformation of each element of NTK and the covariance is not independent. To circumvent this barrier, we derive a unitary quantum aggregation transformation to approximately reconstruct the affine transformation. Consider the quantum operation $\mathcal{D}^0_{uu'} : |u\rangle|u'\rangle|0\rangle|0\rangle \rightarrow |u\rangle|u'\rangle \left(\sqrt{P_{uu'}}\,|0, g_{uu'}\rangle + \sqrt{1-P_{uu'}}\,|1, g'_{uu'}\rangle\right)$ that is employed to estimate the inner product of two feature vectors. Define a unitary operator which is used to perform aggregation transformation

$$\mathcal{U} = \sum_{v\in\mathcal{N}(u)\cup\{u\}} \sum_{v'\in\mathcal{N}(u')\cup\{u'\}} |v\rangle\,|v'\rangle\,\langle v|\,\langle v'| \otimes \mathcal{D}^0_{vv'}, \tag{17}$$

which can be generated by introducing conditional quantum evolution [22]. The operation $\otimes$ denotes the tensor product. We apply the $\mathcal{U}$ with Hadamard gates to the given initial state, which is given as

$$\begin{aligned} &H^{\otimes}\mathcal{U}H^{\otimes}|0\rangle^{\otimes}|0\rangle|0\rangle \rightarrow H^{\otimes}\mathcal{U}\sum_{v,v'}|v,v'\rangle\,|0\rangle|0\rangle \\ &\rightarrow H^{\otimes} \sum_{v\in\mathcal{N}(u)\cup\{u\}} \sum_{v'\in\mathcal{N}(u')\cup\{u'\}} |v,v'\rangle \left(\sqrt{P_{vv'}}\,|0,g_{vv'}\rangle + \sqrt{1-P_{vv'}}\,|1,g'_{vv'}\rangle\right) \\ &\rightarrow \sum_{v\in\mathcal{N}(u)\cup\{u\}} \sum_{v'\in\mathcal{N}(u')\cup\{u'\}} \sqrt{P_{vv'}}\,|0\rangle^{\otimes} + \sqrt{\cdot}\,|\text{other}\rangle + \cdots \end{aligned} \tag{18}$$

where $\sqrt{\cdot}\,|\text{other}\rangle$ represents other computational basis states except for $|0\rangle^{\otimes}$ with amplitude $\sqrt{\cdot}$, and the detailed mathematical expression and the scalar for state normalization are omitted since the result of the affine transformation has been embedded into the amplitude of $|0\rangle^{\otimes}$. The $(\cdot)^{\otimes}$ denotes that there could be multiple unitary operations acting on multiple registers, depending on the number of qubits required to encode the classic data. Similar to the inner product estimation, the quantum aggregation transformation can be performed in superposition and the resulting superposition is

$$\begin{aligned} &\frac{1}{\sqrt{nn'}}\sum_{u\in V}\sum_{u'\in V'}|u\rangle|u'\rangle\left(\sqrt{A_{uu'}}\,|0,y_{uu'}\rangle + \sqrt{1-A_{uu'}}\,|1,y'_{uu'}\rangle\right), \\ &\sqrt{A_{uu'}} = \frac{\sum_{v\in\mathcal{N}(u)\cup\{u\}}\sum_{v'\in\mathcal{N}(u')\cup\{u'\}}\sqrt{P_{vv'}}}{|v|\times|v'|}. \end{aligned} \tag{19}$$

The amplitude $\sqrt{A_{uu'}}$ can be encoded into an ancillary register by using Amplitude Estimation (Theorem 3) and Median Evaluation (Theorem 4). The obtained quantum state $\frac{1}{\sqrt{nn'}}\sum_{u\in V}\sum_{u'\in V'}|u\rangle|u'\rangle|\bar{A}_{uu'}\rangle|y_{uu'}\rangle$ whose third register carries the approximate result after aggregation transformation as described in Eq. 5 and Eq. 15, where $|A_{uu'} - \bar{A}_{uu'}| \leq \epsilon$ and $|y_{uu'}\rangle$ is a garbage state. The runtime is $O(\log(nd)log(1/\Delta)/\epsilon)$ and Δ is the proximity defined by the Median Evaluation. Note that $\bar{A}_{uu'}$ is actually the polynomial combination of the element-wise square root of the NTK from the previous layer, thus it is an approximate aggregation transformation. In the experiment, we empirically show that this approximation has a restrictive effect on the performance.

Quantum kernel estimation. For fully-connected neural network, the calculation of each element of the NTK and its covariance is only reliant on the element at the same position of the covariance matrix in the previous fully-connected layer. Besides, the affine transformation of the GNTK and its covariance can be efficiently approximated by quantum aggregation transformation and the result has been embedded into the basis states of a superposition. In general, there exits a unitary $V : \sum_x |x,0\rangle \rightarrow \sum_x |x, f(x)\rangle$ for any classical function f with the same time complexity to evaluate each element of the NTK and each element of the covariance [48, 71]. Specifically, an oracle which operates as the same as classical function defined by Eq. 8 is implemented on the third register of $\frac{1}{\sqrt{nn'}}\sum_{u\in V}\sum_{u'\in V'}|u\rangle|u'\rangle|\bar{A}_{uu'}\rangle|y_{uu'}\rangle$. The resulting NTK is $\frac{1}{\sqrt{nn'}}\sum_{u\in V}\sum_{u'\in V'}|u\rangle|u'\rangle|\bar{\mathbf{\Theta}}_{uu'}\rangle|y_{uu'}\rangle$, where $\bar{\mathbf{\Theta}}_{uu'}$ is the approximate estimation of its classical counterpart after R fully-connected layers. The oracle is expected to be with the same complexity of its classical counterpart, which is associative to the number of fully-connected layers and is independent on the number of training samples n. For estimation of the GNTK after a transformer layer (Eq. 14), the covariance $\mathbf{\Sigma}^l_R(G,G)$ for any $G \in \mathcal{G}$ requires to be estimated in advance. It means that the state $\frac{1}{n}\sum_{u\in V}\sum_{u'\in V}|u\rangle|u'\rangle|\bar{\mathbf{\Sigma}}_{uu'}\rangle|y_{uu'}\rangle$ must be estimated for any $G(V,E) \in \mathcal{G}$ before input the different graphs, and we only consider the element when $u = u'$. By taking the partial trace on the second register, we obtain the state $\frac{1}{\sqrt{n}}\sum_{u\in V}|u\rangle|\bar{\mathbf{\Sigma}}_{uu}\rangle|y_{uu}\rangle$ for graph G and

$\frac{1}{\sqrt{n'}}\sum_{u'\in V'}|u'\rangle|\bar{\mathbf{\Sigma}}_{u'u'}\rangle|y_{u'u'}\rangle$ for graph G'. Thus, estimation of the GNTK and its covariance corresponding to the multiplication part in Eq. 14 is given as

$$\begin{aligned}\frac{1}{\sqrt{nn'}}\sum_{u\in V}\sum_{u'\in V'}|u\rangle|u'\rangle|\bar{\mathbf{\Theta}}_{uu'}\rangle|y_{uu'}\rangle &\rightarrow \frac{1}{\sqrt{nn'}}\sum_{u\in V}\sum_{u'\in V'}|u\rangle|u'\rangle|\bar{\mathbf{\Theta}}_{uu'}\times\bar{\mathbf{\Sigma}}_{uu}\times\bar{\mathbf{\Sigma}}_{u'u'}\rangle|y_{uu'}\rangle,\\ \frac{1}{\sqrt{nn'}}\sum_{u\in V}\sum_{u'\in V'}|u\rangle|u'\rangle|\bar{\mathbf{\Sigma}}_{uu'}\rangle|y_{uu'}\rangle &\rightarrow \frac{1}{\sqrt{nn'}}\sum_{u\in V}\sum_{u'\in V'}|u\rangle|u'\rangle|\bar{\mathbf{\Sigma}}_{uu'}\times\bar{\mathbf{\Sigma}}_{uu}\times\bar{\mathbf{\Sigma}}_{u'u'}\rangle|y_{uu'}\rangle.\end{aligned}\tag{20}$$

This is performed by using the conditional quantum adder and the multiplier conditioned on the index register, i.e. $|u\rangle$ and $|u'\rangle$, which are designed by [63, 54, 41]. The final GNTK after the transformer layer can be directly generated by additional quantum arithmetic operations that perform an element-wise addition between the covariance to the GNTK.

Estimation the GNTK for multiple layers. The quantum aggregation transformation requires that the approximate NTK and its covariance are embedded into the amplitudes of a superposition. However, after the quantum kernel estimation, these matrix are embedded into the quantum basis states of a superposition. To extract them back to the amplitudes, we apply Conditional Rotation [37] on the register containing the approximate GNTK (and the covariance), which is given by

$$\begin{aligned}\frac{1}{\sqrt{nn'}}\sum_{u\in V}\sum_{u'\in V'}|u\rangle|u'\rangle|\bar{\mathbf{\Theta}}_{uu'}\rangle &\rightarrow \frac{1}{\sqrt{nn'}}\sum_{u\in V}\sum_{u'\in V'}|u\rangle|u'\rangle(a_{uu'}|0\rangle+\sqrt{1-a_{uu'}^2}|1\rangle),\\ \frac{1}{\sqrt{nn'}}\sum_{u\in V}\sum_{u'\in V'}|u\rangle|u'\rangle|\bar{\mathbf{\Sigma}}_{uu'}\rangle &\rightarrow \frac{1}{\sqrt{nn'}}\sum_{u\in V}\sum_{u'\in V'}|u\rangle|u'\rangle(b_{uu'}|0\rangle+\sqrt{1-b_{uu'}^2}|1\rangle),\end{aligned}\tag{21}$$

where $a_{uu'}=\sqrt{\frac{\bar{\mathbf{\Theta}}_{uu'}}{\max_{uu'}(\bar{\mathbf{\Theta}}_{uu'})}}$ and $b_{uu'}=\sqrt{\frac{\bar{\mathbf{\Sigma}}_{uu'}}{\max_{uu'}(\bar{\mathbf{\Sigma}}_{uu'})}}$. We denote this quantum operation as $\mathcal{D}^l, l\in\{1,\ldots,L\}$, where $\mathcal{D}^L$ is used for the quantum readout operation. Similar to the operation $\mathcal{D}^0$, the quantum aggregation transformation can be performed by generating a unitary operator by introducing conditional quantum evolution. Notice that $a_{uu'}$ and $b_{uu'}$ can be viewed as $\sqrt{P_{uu'}}$ in the setting of the single-layer GraphQNTK.

Quantum readout. The resulting NTK is embedded into the basis states of a superposition since the algorithm ends up in the fully-connected layers. Similar to the classic readout operation, the summation of all the elements of the NTK matrix at the L-th layer is required. We use Conditional Rotation to extract the NTK back to the amplitude, and define a unitary $\mathcal{O}$ which is a generalization of the unitary $\mathcal{U}$, where

$$\mathcal{O}=\sum_{v\in V}\sum_{v'\in V}|v\rangle|v'\rangle\langle v|\langle v'|\otimes\mathcal{D}_{vv'}^L.\tag{22}$$

The unitary $\mathcal{O}$ sums the square root of all the elements of the GraphQNTK matrix. And the resulting GraphQNTK between two input graphs is $\bar{\mathbf{\Theta}}_{GG'}=\frac{(\sum_{u\in V,u'\in V'}\sqrt{\bar{\mathbf{\Theta}}_{uu'}})^2}{n\times n'}$, where $\bar{\mathbf{\Theta}}_{uu'}$ is the GraphQNTK of the last layer.

Quantum inference to unseen data. We assume that the test data and the label of the training set are already encoded into the QRAM such that $|\mathbf{k}_*\rangle\in\mathbb{R}^N$, the GraphQNTK between the test graph G^*, can be evaluated as the same way to the evaluation between the training data. Let $\bar{\mathbf{\Theta}}\in\mathbb{R}^{N\times N}$ denote the GraphQNTK. The final prediction for a test datapoint G_* is

$$f^*(G^*)=\langle\mathbf{k}_*|\bar{\mathbf{\Theta}}^{-1}|\mathbf{y}\rangle,\tag{23}$$

which requires solving the linear equation $|\mathbf{E}\rangle=\bar{\mathbf{\Theta}}^{-1}|\mathbf{y}\rangle$ and performing inner product estimation on $\langle\mathbf{k}_*|\mathbf{E}\rangle$. A popular quantum algorithm which is designed to solve the quantum linear systems problem (QLSP) is developed by [13], and its runtime is $O(\log(N)\kappa s$ polylog $(\kappa s/\epsilon))$ where s is the sparsity of matrix $\bar{\mathbf{\Theta}}$ and κ is the condition number. To realize the quantum speedup, we assume a specific sparsity pattern is created in the quantum storage that only keeps $O(\log N)$ number of non-zero elements of the $N\times N$ GraphQNTK matrix and the well-conditioning is achieved by using Gershgorin circle theorem similar to [71].

2.4 Complexity Study

In Sec. 2.3, we discuss how to approximately estimate GNTK using quantum computing paradigm between two input graphs. The time complexity is dominated by the quantum aggregation transformation procedure as it requires encoding the amplitude into an additional register, which takes

Table 1: Classification accuracies on graphs with discrete node attributes. The AttentionGNTK denotes the GNTK with attention mechanism without both sparsity and well conditioning, while the GraphQNTK is the kernel after performing these two transformations to meet the conditions for the use of quantum matrix inversion. The results of other models are taken from [17] except QS-CNN, which we evaluate on our dataset separation.

Dataset	MUTAG	PROTEINS	PTC	NCI1	IMDB-B	IMDB-M
WL subtree [57]	90.4 ± 5.7	75.0 ± 3.1	59.9 ± 4.3	**86.0 ± 1.8**	73.8 ± 3.9	50.9 ± 3.8
AWL [30]	87.9 ± 9.8	-	-	-	74.5 ± 5.9	51.5 ± 3.6
RetGK [69]	90.3 ± 1.1	75.8 ± 0.6	62.5 ± 1.6	84.5 ± 0.2	71.9 ± 1.0	47.7 ± 0.3
GNTK [17]	90.0 ± 8.5	75.6 ± 4.2	**67.9 ± 6.9**	84.2 ± 1.5	76.9 ± 3.6	52.8 ± 4.6
GCN [39]	85.6 ± 5.8	76.0 ± 3.2	64.2 ± 4.3	80.2 ± 2.0	74.0 ± 3.4	51.9 ± 3.8
GraphSAGE [21]	85.1 ± 7.6	75.9 ± 3.2	63.9 ± 7.7	77.7 ± 1.5	72.3 ± 5.3	50.9 ± 2.2
PatchySAN [49]	92.6 ± 4.2	75.9 ± 2.8	60.0 ± 4.8	78.6 ± 1.9	71.0 ± 2.2	45.2 ± 2.8
GIN [67]	89.4 ± 5.6	76.2 ± 2.8	64.6 ± 7.0	82.7 ± 1.7	75.1 ± 5.1	52.3 ± 2.8
QS-CNN [70]	**93.1 ± 4.7**	**78.2 ± 4.6**	66.0 ± 4.4	81.4 ± 2.6	72.1 ± 3.7	46.2 ± 4.2
AttentionGNTK	90.0 ± 8.5	76.2 ± 3.8	66.2 ± 5.1	84.1 ± 1.2	**76.9 ± 3.2**	**52.9 ± 3.5**
GraphQNTK	88.4 ± 6.5	71.1 ± 3.2	62.9 ± 5.0	77.2 ± 2.7	73.3 ± 3.6	48.1 ± 4.3

$O(\log(nd)log(1/\Delta)/\epsilon)$ time. Other quantum operations including estimation of the inner product, estimation of the GNTK within the neighborhood aggregation and the fully-connected feature updating and quantum readout are totally unitary operations which can be efficiently performed under the regime of quantum computing. For estimating GNTK of each pairs of the graphs (G, G') where $G, G' \in \mathcal{G}$, each element of GraphQNTK $\bar{\mathbf{\Theta}}$ can be generated simultaneously by introducing auxiliary index registers. The quantum runtime is $O(\log(Nnd))$. However, evaluating GNTK of the infinite-width-limit attention requires computing the kernel where the input is two same graphs, which can be implemented in time $O(N)$. The result should be stored in QRAM in advance which will be used to update GNTK corresponding the multi-head attention as described in Eq. 14. Therefore, it takes $O(N\log(Nnd))$ time to train the proposed quantum graph learning model, which achieves quadratic speedup compared to the existing GKs and completed approaches with $O(N^2)$ time.

3 Experiments

We evaluate our method for both GNTK and GraphQNTK with attention mechanism on several graph classification datasets involving either discrete or continuous attributes. All the experiments are performed on a workstation with a single machine with 1TB memory, one physical CPU with 28 cores Intel(R) Xeon(R) W-3175X CPU @ 3.10GHz, and a single GPU (Nvidia Quadro RTX 8000). For our method and all the compared models, We follow the same setting as [17, 67], and report the average test accuracy and its standard deviation over a 10-fold cross validation on each dataset.

3.1 Experiments Setup

Datasets. For graph with discrete attributes, the benchmark datasets include four bioinformatics datasets MUTAG, PTC, NCI1, PROTEINS and three social network datasets IMDB-BINARY, IMDB-MULTI. For each graph, the input attributes is category of the node and they are transformed to one-hot encoding representations. For datasets where the graphs have no node features, i.e. only graph structure matters, we use degrees as input node features. For graph with continuous attributes, we selcect four benchmark datasets including ENZYMES, PROTEINS full, BZR, COX2. All the datasets can be found in [38]. The statistic information of the datasets are given in Tab. 3 in Appendix.

Compared baselines. We compare our method with state-of-the-art GKs such as WL kernel [57], AWL [30], RetGK [69], GNTK [17], WWL [62], and GNNs including GCN [39], PatchySAN [49], GCKN [10], GraphSAGE [21] and GIN [67]. For quantum graph learning, there are very few baseline available. We report the performance of the quantum walk based subgraph convolutional neural network (QS-CNN) developed by [70]. The data separation we use is the same as [67] for graph

Table 2: Classification accuracies on graphs with continuous attributes. The accuracies of other models are taken from [10]. We only take the results of GCKN under the supervised learning for a fail comparison. We utilize the similar settings that preprocess the continuous node features to a normalized feature vector as in [62] for fair comparison (Note that the data encoded into the QRAM requires normalization, thus it is reasonable to use this data-prepossessing operation).

Dataset	ENZYMES	PROTEINS	BZR	COX2
RBF-WL [62]	68.4 ± 1.5	75.4 ± 0.3	81.0 ± 1.7	75.5 ± 1.5
HGK-WL [47]	63.0 ± 0.7	75.9 ± 0.2	78.6 ± 0.6	78.1 ± 0.5
HGK-SP [47]	66.4 ± 0.4	75.8 ± 0.2	76.4 ± 0.7	72.6 ± 1.2
WWL [62]	**73.3** ± **0.9**	**77.9** ± **0.8**	84.4 ± 2.0	78.3 ± 0.5
GNTK [17]	69.6 ± 0.9	75.7 ± 0.2	85.5 ± 0.8	79.6 ± 0.4
GCKN [10]	72.8 ± 1.0	77.6 ± 0.4	86.4 ± 0.5	81.7 ± 0.7
AttentionGNTK	69.2 ± 1.1	76.8 ± 1.2	**86.7** ± **1.3**	**82.1** ± **0.4**
GraphQNTK	64.8 ± 0.7	72.5 ± 0.3	80.1 ± 1.7	74.3 ± 1.9

datasets with discrete attributes. For graph dataset with continuous attributes, we follow the same protocol as used in [62] to normalize the input feature vectors for a fair comparison.

3.2 Experiment Results

We apply different hyper-parameter settings to $L \in \{2, 4, 6, 8\}$ and $R \in \{1, 2, 3\}$ and select the model with the best averaged accuracy. We test the kernel regression using SVM classifier and the regularization parameter is determined using the search protocol which is the same as the [17]. We report the performance of the quantum approximate GNTK before and after the matrix sparsity and conditioning operations. The numerical results are listed in Tab. 1 for datasets with discrete attributes and Tab. 2 for datasets with continuous attributes. The attention method we integrate to the infinite-width GNNs brings to an improvement in the performance of the model. The results show that the GNTK with attention mechanism achieves better classification accuracy for graph data with medium number of nodes and edges. It is demonstrated that the infinite-width-limit attention captures global node similarity semantics and learns effective structure of the provided graph, which brings an remarkable accuracy improvement of the model compared with the vanilla GNTK [17]. Moreover, our model performs better than QS-CNN on more than 60% of the datasets with discrete attributes, given the caveat that QS-CNN is a hybrid graph learning model where the contribution of the classic components (CNNs, spatial message passing) in their model cannot be ignored. While the matrix sparsity and conditioning operations have a great influence on the model's performance, it can be found that the classification performance of GNTK evaluated by quantum algorithms is still comparable with that of GKs and vanilla GNNs, where a tradeoff exists between the performance of the model and the quantum computational efficiency.

4 Conclusion and Broader Impact

This paper has presented a quantum graph learning model to characterize the structural information while using quantum parallelism to improve computing efficiency. We propose quantum algorithm to approximately estimate the neural tangent kernel of the underlying graph neural network where a multi-head quantum attention mechanism is introduced to incorporate semantic similarity of nodes. Empirical results on graph classification tasks as well as theoretical analysis show the superiority of our method. The limitation of the paper is that currently it only addresses graph-level embedding and we leave node-level quantum learning for future work. Our work may raise concerns for encryption, privacy protection etc. when the quantum hardware become more feasible.

Acknowledgement

This work was partly supported by National Key Research and Development Program of China (2020AAA0107600), National Natural Science Foundation of China (61972250, 72061127003), and Shanghai Municipal Science and Technology (Major) Project (2021SHZDZX0102, 22511105100).

References

[1] Xing Ai, Zhihong Zhang, Luzhe Sun, Junchi Yan, and Edwin Hancock. Decompositional quantum graph neural network. *arXiv:2201.05158*, 2022.

[2] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *Advances in Neural Information Processing Systems*, 32, 2019.

[3] Lu Bai, Luca Rossi, Andrea Torsello, and Edwin R Hancock. A quantum jensen–shannon graph kernel for unattributed graphs. *Pattern Recognition*, 2015.

[4] Lu Bai, Luca Rossi, Lixin Cui, Zhihong Zhang, Peng Ren, Xiao Bai, and Edwin Hancock. Quantum kernels for unattributed graphs using discrete-time quantum walks. *Pattern Recognition Letters*, 2017.

[5] Lu Bai, Yuhang Jiao, Lixin Cui, Luca Rossi, Yue Wang, Philip Yu, and Edwin Hancock. Learning graph convolutional networks based on quantum vertex information propagation. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[6] Johannes Bausch. Recurrent quantum neural networks. *Advances in Neural Information Processing Systems*, 33:1368–1379, 2020.

[7] Kerstin Beer, Megha Khosla, Julius Köhler, and Tobias J Osborne. Quantum machine learning of graph-structured data. *arXiv:2103.10837*, 2021.

[8] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 2017.

[9] Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002.

[10] Dexiong Chen, Laurent Jacob, and Julien Mairal. Convolutional kernel networks for graph-structured data. In *International Conference on Machine Learning*, pages 1576–1586. PMLR, 2020.

[11] Samuel Yen-Chi Chen, Tzu-Chieh Wei, Chao Zhang, Haiwang Yu, and Shinjae Yoo. Hybrid quantum-classical graph convolutional network. *arXiv:2101.06189*, 2021.

[12] Yu Chen, Lingfei Wu, and Mohammed Zaki. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. *Advances in Neural Information Processing Systems*, 33:19314–19326, 2020.

[13] Andrew M Childs, Robin Kothari, and Rolando D Somma. Quantum algorithm for systems of linear equations with exponentially improved dependence on precision. *SIAM Journal on Computing*, 46(6):1920–1950, 2017.

[14] Iris Cong, Soonwon Choi, and Mikhail D Lukin. Quantum convolutional neural networks. *Nature Physics*, 2019.

[15] Amit Daniely, Roy Frostig, and Yoram Singer. Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity. *Advances In Neural Information Processing Systems*, 29, 2016.

[16] Stefan Dernbach, Arman Mohseni-Kabir, Siddharth Pal, and Don Towsley. Quantum walk neural networks for graph-structured data. In *Complex Networks and Their Applications*, 2018.

[17] Simon S Du, Kangcheng Hou, Russ R Salakhutdinov, Barnabas Poczos, Ruosong Wang, and Keyulu Xu. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. *Advances in Neural Information Processing Systems*, 32, 2019.

[18] Jinyuan Fang, Shangsong Liang, Zaiqiao Meng, and Qiang Zhang. Gaussian process with graph convolutional kernel for relational learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 353–363, 2021.

[19] Iulia Georgescu. Quantum enhancement in generative models. *Nature Reviews Physics*, 4(6): 362–362, 2022.

[20] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[21] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, 30, 2017.

[22] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 2009.

[23] David Haussler. Convolution kernels on discrete structures. Technical report, Department of Computer Science, University of California, 1999.

[24] Vojtěch Havlíček, Antonio D Córcoles, Kristan Temme, Aram W Harrow, Abhinav Kandala, Jerry M Chow, and Jay M Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209–212, 2019.

[25] Louis-Paul Henry, Slimane Thabet, Constantin Dalyac, and Loïc Henriet. Quantum evolution kernel: Machine learning on graphs with programmable arrays of qubits. *Physical Review A*, 2021.

[26] Jiri Hron, Yasaman Bahri, Jascha Sohl-Dickstein, and Roman Novak. Infinite attention: Nngp and ntk for deep attention networks. In *International Conference on Machine Learning*, pages 4376–4386. PMLR, 2020.

[27] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *Proceedings of The Web Conference 2020*, pages 2704–2710, 2020.

[28] Hsin-Yuan Huang, Michael Broughton, Masoud Mohseni, Ryan Babbush, Sergio Boixo, Hartmut Neven, and Jarrod R McClean. Power of data in quantum machine learning. *Nature communications*, 12(1):1–9, 2021.

[29] Wei Huang, Yayong Li, weitao Du, Richard Xu, Jie Yin, Ling Chen, and Miao Zhang. Towards deepening graph neural networks: A GNTK-based optimization perspective. In *International Conference on Learning Representations*, 2022.

[30] Sergey Ivanov and Evgeny Burnaev. Anonymous walk embeddings. In *International conference on machine learning*, pages 2186–2195. PMLR, 2018.

[31] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

[32] Ashish Kapoor, Nathan Wiebe, and Krysta Svore. Quantum perceptron models. *Advances in neural information processing systems*, 29, 2016.

[33] Julia Kempe. Quantum random walks: an introductory overview. *Contemporary Physics*, 2003.

[34] Iordanis Kerenidis and Jonas Landman. Quantum spectral clustering. *Physical Review A*, 103 (4):042415, 2021.

[35] Iordanis Kerenidis and Anupam Prakash. Quantum recommendation systems. In *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[36] Iordanis Kerenidis, Jonas Landman, Alessandro Luongo, and Anupam Prakash. q-means: A quantum algorithm for unsupervised machine learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[37] Iordanis Kerenidis, Jonas Landman, and Anupam Prakash. Quantum algorithms for deep convolutional neural networks. In *International Conference on Learning Representations*, 2020.

[38] Kristian Kersting, Nils M. Kriege, Christopher Morris, Petra Mutzel, and Marion Neumann. Benchmark data sets for graph kernels, 2016. URL `http://graphkernels.cs.tu-dortmund.de`.

[39] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[40] Tao Lei, Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Deriving neural architectures from sequence and graph kernels. In *International Conference on Machine Learning*, pages 2024–2033. PMLR, 2017.

[41] Hai-Sheng Li, Ping Fan, Haiying Xia, Huiling Peng, and Gui-Lu Long. Efficient quantum arithmetic operation circuits for quantum image processing. *SCIENCE CHINA Physics, Mechanics & Astronomy*, 63(8):1–13, 2020.

[42] Junyu Liu, Francesco Tacchino, Jennifer R Glick, Liang Jiang, and Antonio Mezzacapo. Representation learning via quantum neural tangent kernels. *arXiv preprint arXiv:2111.04225*, 2021.

[43] Yunchao Liu, Srinivasan Arunachalam, and Kristan Temme. A rigorous and robust quantum speed-up in supervised machine learning. *Nature Physics*, 2021.

[44] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum algorithms for supervised and unsupervised machine learning. *arXiv preprint arXiv:1307.0411*, 2013.

[45] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum principal component analysis. *Nature Physics*, 10(9):631–633, 2014.

[46] Péter Mernyei, Konstantinos Meichanetzidis, and Ismail Ilkan Ceylan. Equivariant quantum graph circuits. In *International Conference on Machine Learning*, pages 15401–15420. PMLR, 2022.

[47] Christopher Morris, Nils M Kriege, Kristian Kersting, and Petra Mutzel. Faster kernels for graphs with continuous attributes via hashing. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 1095–1100. IEEE, 2016.

[48] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

[49] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pages 2014–2023. PMLR, 2016.

[50] Giannis Nikolentzos, Polykarpos Meladianos, Antoine J-P Tixier, Konstantinos Skianis, and Michalis Vazirgiannis. Kernel graph convolutional neural nets, 2018.

[51] Roman Novak, Lechao Xiao, Yasaman Bahri, Jaehoon Lee, Greg Yang, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-dickstein. Bayesian deep convolutional networks with many channels are gaussian processes. In *International Conference on Learning Representations*, 2019.

[52] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. *Physical review letters*, 2014.

[53] Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. Self-supervised graph transformer on large-scale molecular data. *Advances in Neural Information Processing Systems*, 33:12559–12571, 2020.

[54] Lidia Ruiz-Perez and Juan Carlos Garcia-Escartin. Quantum arithmetic with the quantum fourier transform. *Quantum Information Processing*, 16(6):1–14, 2017.

[55] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. An introduction to quantum machine learning. *Contemporary Physics*, 56(2):172–185, 2015.

[56] Maria Schuld, Kamil Brádler, Robert Israel, Daiqin Su, and Brajesh Gupt. Measuring the similarity of graphs with a gaussian boson sampler. *Physical Review A*, 2020.

[57] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.

[58] Kazuya Shimizu and Ryuhei Mori. Exponential-time quantum algorithms for graph coloring problems. In *Latin American Symposium on Theoretical Informatics*, pages 387–398. Springer, 2021.

[59] Norihito Shirai, Kenji Kubo, Kosuke Mitarai, and Keisuke Fujii. Quantum tangent kernel. *arXiv preprint arXiv:2111.02951*, 2021.

[60] Francesco Tacchino, Chiara Macchiavello, Dario Gerace, and Daniele Bajoni. An artificial neuron implemented on an actual quantum processor. *npj Quantum Information*, 5(1):1–8, 2019.

[61] Yehui Tang, Junchi Yan, and Hancock Edwin. From quantum graph computing to quantum graph learning: A survey. *arXiv preprint arXiv:2202.09506*, 2022.

[62] Matteo Togninalli, Elisabetta Ghisu, Felipe Llinares-López, Bastian Rieck, and Karsten Borgwardt. Wasserstein weisfeiler-lehman graph kernels. *Advances in Neural Information Processing Systems*, 32, 2019.

[63] Vlatko Vedral, Adriano Barenco, and Artur Ekert. Quantum networks for elementary arithmetic operations. *Physical Review A*, 54(1):147, 1996.

[64] Guillaume Verdon, Trevor McCourt, Enxhell Luzhnica, Vikash Singh, Stefan Leichenauer, and Jack Hidary. Quantum graph neural networks. *arXiv:1909.12264*, 2019.

[65] Nathan Wiebe, Ashish Kapoor, and Krysta M Svore. Quantum algorithms for nearest-neighbor methods for supervised and unsupervised learning. *Quantum Information & Computation*, 15 (3-4):316–356, 2015.

[66] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[67] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[68] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34, 2021.

[69] Zhen Zhang, Mianzhi Wang, Yijian Xiang, Yan Huang, and Arye Nehorai. Retgk: Graph kernels based on return probabilities of random walks. *Advances in Neural Information Processing Systems*, 31, 2018.

[70] Zhihong Zhang, Dongdong Chen, Jianjia Wang, Lu Bai, and Edwin R Hancock. Quantum-based subgraph convolutional neural networks. *Pattern Recognition*, 2019.

[71] Alexander Zlokapa, Hartmut Neven, and Seth Lloyd. A quantum algorithm for training wide and deep classical neural networks. *arXiv:2107.09200*, 2021.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.

- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Related Works

In this section, we provide background on quantum graph learning and graph neural networks that have the potential to be trained using quantum computing.

A.1 Graph Kernel Neural Network

Graph kernel neural networks [17, 10, 18] is a class of graph learning method combining the properties of both GNNs and GKs. Forward process of the model tends to transmit node information like GNNs [66], layer by layer, whereas the node (or graph) features live in the implicit reproducing kernel Hilbert space (RKHS) of a specific kernel [40]. The mainstream of graph kernel neural networks

Table 3: Statistic information of the used datasets.

Dataset	MUTAG	PROTEINS	PTC	NCI1	IMDB-B	IMDB-M	ENZYMES	BZR	COX2
size	188	1113	344	4110	1000	1500	600	405	467
classes	2	2	2	2	2	3	6	2	2
attr. dim.	-	-	-	-	-	-	18	3	3
avg. nodes	18	39	26	30	20	13	32.6	35.8	41.2
avg. edges	20	73	51	32	97	66	62.1	38.3	43.5

can be divided into completed and approximated approaches. For the former, the output is a kernel matrix where each entry denotes the similarity of graph pairs, afterwards support vector machin is used to perform classification or regression task. While the later generates the approximate feature of the finite projected RKHS at the expense of information loss.

We consider the completed approaches for the basis of our proposed quantum graph learning model, since the access to the explicit feature information requires measuring the relative quantum representations, which incurs quantum collapse [33]. In the next section, we demonstrate that the graph tangent kernel neural networks coincides with the condition of quantum parallel implementation by introducing the quantum aggregation transformation and the quantum kernel estimation techniques.

A.2 Quantum Graph Learning

Quantum graph learning aims at leveraging quantum physics to extract graph structural information, bringing up new possibilities for quantum computing applications. It is generally nontrivial to analyze classical data under the regime of quantum computing, since the encoding and decoding between classical vectors (or matrices) and their corresponding quantum states should be carefully designed. In addition, encoding the irregular graph data and diverse structure topology may incur different configurations of quantum models. Advanced contributions has developed some techniques to overcome these issues. A hybrid graph learning method developed by [70, 16] encode the structure information and generate a new adjacent matrix evaluating by the using quantum walk. The resulting adjacent information captures the global topological arrangement information for graph substructures. Adiabatic evolution [64] and conditional unitary [46] are applied to evolve the quantum systems dependent on the underlying graph structure. In addition, the node attribute is encoded using variational circuit [1] or a quantum random access memory [58]. Processing the encoded quantum representation of the original graph can be realized via either a naive quantum algorithm [64] or a hybrid method [11]. Then a post-processing operation is performed to further analyze the quantum output. A brief review about quantum graph learning is illustrated in Fig. 1. Generally speaking, the researchers exploit to encode the graph structure and node features in the quantum system through various schemes, and then process the information through quantum layers and auxiliary classical layers. Finally, the (quantum) results are decoded through post-processing. However, most quantum graph learning models requires that adjustable parameters in the quantum algorithm need to be updated frequently, where takes great computational overheads. Moreover, the classical components in post-processing may dominate the performance of the model, thus weakening the role of the quantum part. In this paper, We seek to establish a parameter-free quantum graph learning model to maximize the efficacy of quantum computing.

We notice that there are researches which are abbreviated as QNTK [59, 42], similar to ours nominally. But their definition is quite different from ours. The motivation of these two papers is to analyze the trainability and expressive power of variational quantum circuits through NTK. In contrast, in our work, QNTK is a metric measuring the similarity of two input graphs. In this context, NTK is the kernel that captures the dynamics of infinite-width GNNs, as well as the multi-head attention where the number of heads and the dimension of output go to infinity.

B More Analysis

B.1 Quantum Access Memory

Theorem 1 *Let* $|\mathbf{X}_p\rangle = \frac{1}{\|\mathbf{X}_p\|}\sum_{q=0}^{d-1}\mathbf{X}_{pq}|j\rangle$ *denotes the amplitude encoding of the* p*-th row of data* $\mathbf{X} \in \mathbb{R}^{n\times d}$*. There exists a data structure to store the entries of* $\mathbf{X}$ *into the QRAM which is stated as* *i)* $|p\rangle\,|0\rangle \to |p\rangle\,|\mathbf{X}_p\rangle$

ii) $|0\rangle \to \frac{1}{\|\mathbf{X}\|_F}\sum_p \|\mathbf{X}_p\|\,|p\rangle$

in time T for $p \in [n]$. Using the binary tree QRAM architecture proposed by [35], the time T to store and readout a new element scale logarithmically with respect to both n and d.

B.2 Inner Product Estimation

Theorem 2 *There exists a quantum operation $\mathcal{A}$ that evaluates the inner product of two quantum representations with respect to their d-dimensional classical vectors in time $O(\log d)$.*

Proof. *By introducing an auxiliary register, with the initial state $|p\rangle|q\rangle\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|0\rangle$, the map $\frac{1}{\sqrt{2}}(|p\rangle|q\rangle|0\rangle|0\rangle + |p\rangle|q\rangle|1\rangle|0\rangle) \to \frac{1}{\sqrt{2}}(|p\rangle|q\rangle|0\rangle|\,|\mathbf{X}_p\rangle\rangle + |p\rangle|q\rangle|1\rangle|\,|\mathbf{X}_q\rangle\rangle)$ can be performed in $O(\log d)$ for two quantum representations $|\mathbf{X}_p\rangle$ and $|\mathbf{X}_q\rangle$ with respect to their classical vectors $\mathbf{X}_p \in \mathbb{R}^d$ and $\mathbf{X}_q \in \mathbb{R}^d$. Applying a Hadamard gate on the third register, the state becomes*

$$\frac{1}{2}|p\rangle|q\rangle\left(|0\rangle\left(|\mathbf{X}_p\rangle + |\mathbf{X}_q\rangle\right) + |1\rangle\left(|\mathbf{X}_p\rangle - |\mathbf{X}_q\rangle\right)\right). \tag{24}$$

The probability of measuring 0 on the third register is given by $P_{pq} = \frac{1+\langle\mathbf{X}_p|\mathbf{X}_q\rangle}{2}$. Thus the state defined by Eq. 24 can be reformulated as $|p\rangle|q\rangle\left(\sqrt{P_{pq}}\,|0, g_{pq}\rangle + \sqrt{1-P_{pq}}\,|1, g'_{pq}\rangle\right)$ where $|g_{pq}\rangle$ and $\left|g'_{pq}\right\rangle$ are garbage states.

B.3 Amplitude Estimation

Theorem 3 *Given a unitary operator U such that $U : |0\rangle \mapsto \sqrt{p}|y\rangle|0\rangle + \sqrt{1-p}\,|y'\rangle\,|1\rangle$ in time T, where $p > 0$ is the probability of measuring 0, it is possible to obtain the state $|y\rangle|0\rangle$ using $O(\frac{T}{\sqrt{p}})$ queries to U, or to estimate p with relative error δ using $O(\frac{T}{\delta\sqrt{p}})$ queries to U. The detailed proof can be found in [9].*

B.4 Median Evaluation

Theorem 4 *Consider a unitary $U : |0^{\otimes m}\rangle \mapsto \sqrt{\alpha}|v, 1\rangle + \sqrt{1-\alpha}|g, 0\rangle$ for some $1/2 \le \alpha \le 1$ in time T. There exits a quantum algorithm that, for any $\Delta > 0$ and for any $1/2 < \alpha_0 \le \alpha$, produce a state $|\psi\rangle$ such that $\||\psi\rangle - \left|0^{\otimes mL}\right\rangle|x\rangle\| \le \sqrt{2\Delta}$ for some integer L in time*

$$2T\left\lceil\frac{\log(1/\Delta)}{2\left(|\alpha_0| - 1/2\right)^2}\right\rceil. \tag{25}$$

Refer to [65] for a detailed proof.

C Additional clarification of the proposed model

C.1 Definition of the infinite-width GNN and multi-head attention

GNTK is a generalization of NTK from infinite-width fully connected neural network to graph neural network, which is a well-established approach in recent literature and thus we miss some detailed explanation in our paper for space saving (and also due to the complexity of the details which otherwise will cost a lot of space), and the details can be found in [17]. Here we give an intuitive illustration.

Consider a general GNN with the neighborhood feature aggregation function

$$\hat{\mathbf{h}}_u^l := \sum_{v\in\mathcal{N}(u)\cup\{u\}} \mathbf{h}_v^{(l-1)},$$

and the central node feature update function (R fully connected layers)

$$\mathbf{h}_u^l := \sqrt{\frac{c_\sigma}{d_R^l}}\sigma\left(\mathbf{W}_R^l\sqrt{\frac{c_\sigma}{d_{R-1}^l}}\sigma\left(\mathbf{W}_{R-1}^l\cdots\sqrt{\frac{c_\sigma}{d_1^l}}\cdot\sigma\left(\mathbf{W}_1^l\hat{\mathbf{h}}_u^l\right)\right)\right),$$

where $\mathbf{h}_u^l$ denotes the feature vector of node u in the l-th layer, $\mathcal{N}(u)$ denotes the neighborhood of node u, $\mathbf{W}_r^l \in \mathcal{R}^{d_r^l \times d_{r+1}^l}$ is the weight matrix ($r = 1, \dots, R$), c_σ is a scaling factor. The infinite-width of GNN means that the output dimension d_r^l of $\mathbf{W}_r^l$ goes to infinity for $r = 1, \dots, R$ and $l = 1, \dots, L$.

Apart from the infinite-width GNN, the infinite-width transformer, which is exploited by us to enhance the GNTK, has also been investigated in [26]. Here we briefly explain what an infinite-width transformer looks like. Consider a single attention layer

$$\mathbf{Q}^h = \mathbf{H}\mathbf{W}_Q^h, \mathbf{K}^h = \mathbf{H}\mathbf{W}_K^h, \mathbf{V}^h = \mathbf{H}\mathbf{W}_V^h,$$

$$\mathbf{G}^h = \frac{\mathbf{Q}^h {\mathbf{K}^h}^\top}{\sqrt{s}}, \widehat{\mathbf{H}}^h = \zeta\left(\mathbf{G}^h\right)\mathbf{V}^h,$$

in Eq. 10, where $\mathbf{H} \in \mathcal{R}^{n \times d}$ is the node feature matrix and we ignore the superscript and the subscript for simplicity. $\mathbf{Q}^h$, $\mathbf{K}^h$, $\mathbf{V}^h \in \mathcal{R}^{d \times d'}$ are weight matrices corresponding to Query, Key and Value, respectively. s is a scaling factor. The square matrix $\mathbf{G}^h \in \mathcal{R}^{n \times n}$ can be viewed as a matrix whose element corresponds to the similarity of each pair of nodes. Then the operation $\widehat{\mathbf{H}}^h = \zeta\left(\mathbf{G}^h\right)\mathbf{V}^h$ transforms the node features of the last layer to the next layer depending on the node similarity. For a multi-head attention (transformer) layer, the equation is given as

$$transformer(\mathbf{H}) = concat(\widehat{\mathbf{H}}^1 || \widehat{\mathbf{H}}^2 || \cdots || \widehat{\mathbf{H}}^H)\mathbf{W}_O,$$

where $\mathbf{W}_O \in \mathcal{R}^{Hd' \times d''}$ is the weight matrix. The infinite-width transformer means that the output dimension d' of $\mathbf{Q}^h$, $\mathbf{K}^h$, $\mathbf{V}^h$, the output dimension d'' of $\mathbf{W}_O$, and the number of heads H go to infinity.

C.2 Our contributions beyond the quantum speedup

Although the current scale of quantum hardware can not support the application of large-scale quantum algorithms, it still can not stop the widespread attention of academia and industry to quantum computing, especially in quantum machine learning [46, 19]. We hope that our model can provide technical guidance for future research, and better exploit the immense benefits of quantum computing even in a classic simulation condition.

Apart from quantum speedup, there are additional contributions of our proposed model, which have been illustrated in Line 66 to Line 80 in the introduction section. And we hope that this novel graph learning model can bridge the gap between graph neural tangent kernel methods and attention mechanism. Here we briefly give an illustration.

Better performance compared with state-of-art graph models. The numerical results in Tab. 1 and Tab. 2 demonstrate that our model (AttentionGNTK) outperforms GNTK on 7 out of 10 datasets, reflecting that the introduced infinite-width multi-head attention is useful and can better capture distinct properties between different graphs. Furthermore, we compare our model with another quantum-inspired graph learner QS-CNN. Our model surpasses QS-CNN on 4 out of 6 datasets and performs similarly on the rest 2 datasets. It shows the superiority of our model in quantum graph learning.

More robust compared with GNNs when the number of layers becomes larger. As illustrated in Tab. 5, when the number of layers is larger than 5 (but less than 10), Our model is more resistant to the oversmooth problem than GIN.

Fewer layers for reaching the peak of classification accuracy compared with GNTK. As illustrated in Tab. 6, our model (attentionGNTK) reaches the peak of classification accuracy when the number of layers is small, while GNTK needs more layers to reach, indicating that our model is easier to capture the global structure information of the graph.

Table 4: Running time comparison between different models.

	MUTAG	NCI1	IMDB-B	IMDB-M
GIN	22 sec	67 min	19 min	24 min
GNTK	9 sec	18 min	4 min	7 min
Ours	14 sec	21 min	4 min	9 min

Table 5: Classification accuracy between GIN [67] and ours with respect to different layers.

	GIN(4)	Ours(4)	GIN(6)	Ours(6)	GIN(8)	Ours(8)
MUTAG	87.6 ± 6.2	89.1 ± 7.8	88.5 ± 5.6	90.0 ± 8.5	86.2 ± 6.4	88.4 ± 7.4
PROTEINS	75.5 ± 3.0	75.0 ± 4.1	74.3 ± 3.0	76.1 ± 3.8	72.8 ± 3.5	74.2 ± 4.4
PTC	62.8 ± 5.0	64.9 ± 5.3	62.0 ± 6.2	66.2 ± 5.1	61.2 ± 7.1	63.4 ± 6.6
NCI1	82.3 ± 3.6	84.1 ± 1.2	80.1 ± 2.4	83.8 ± 1.2	77.2 ± 3.3	82.3 ± 2.2
IMDB-B	73.2 ± 4.1	75.7 ± 2.8	74.4 ± 6.0	76.9 ± 4.3	72.1 ± 5.2	75.1 ± 4.0
IMDB-M	51.7 ± 3.7	52.0 ± 4.1	52.0 ± 2.6	51.9 ± 3.7	48.2 ± 4.3	50.3 ± 4.5

D Additional Experiments and Discussion

D.1 Running time comparison

We supplement the training time of our model on four selected datasets and compare it with two other models. To make a fair comparison, we set the layers of all the models to 2. All the experiments are performed on a workstation with a single machine with 1TB memory, one physical CPU with 28 cores Intel® Xeon® W-3175X CPU @ 3.10GHz, and a single GPU (Nvidia Quadro RTX 8000). The results are shown in Tab. 4. Although the speedup introduced by the quantum algorithm depends on the quantum devices, it shows that our proposed model still has a computational overhead reduction when training on classic computers. The running time is slightly higher than that of GNTK which is a lack of attention mechanism. It is noticed that our model. The runtime of our model is apparently faster than that of GIN.

It is worth mentioning that the quadratic quantum speedup will be realized when the quantum hardware becomes more feasible.

D.2 Model sensitivity to the number of layers

In the main body of the paper, we report the best classification accuracy of the model when the the number of layers L is selected from $\{2, 4, 6, 8\}$. We compare the graph classification accuracy bewteen GIN and our model at the same number of layers and the results are given in Tab. 5 and Tab. 6. The number in parentheses in the table indicates the number of layers.

From Tab. 5, it is shown that our model (AttentionGNTK) is more robust compared with GIN when the number of layers becomes larger. The main reason is that an additional feature aggregation, e.g., the transformer, can slow down the convergence rate, which is consistent with the observations in [29] that connectivity enhancement can help wide and deep GNNs to avoid a discrepancy between prediction and the ground truth.

While in Tab. 6, the results empirically demonstrate that our model (attentionGNTK) reaches the peak of classification accuracy when the number of layers is small, while GNTK needs more layers to reach, indicating that our model is easier to capture the global structure information of the graph. This could be interpreted from the theoretical perspective. The transformer (Eq. 12) captured the semantic information between each pair of (connected and disconnected) nodes with similar features. Consider $\mathbf{G} = \frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{s}}$ in Eq. 10, where $\mathbf{Q}$ and $\mathbf{K}$ are linear transformation of the node feature matrix and we ignore the superscript and the subscript for simplicity. The $\mathbf{G}$ can be viewed as a matrix whose element corresponds to the similarity of each pair of nodes. Then the operation $\widehat{\mathbf{H}} = \zeta\left(\mathbf{G}\right)\mathbf{V}$ transforms the node features of the last layer to the next layer depending on the node similarity. This enables the model to make better use of the graph structure to transmit information and perceive topology information over long distances.

Table 6: Classification accuracy between GNTK [17] and ours with respect to different layers.

	GNTK(4)	Ours(4)	GNTK(6)	Ours(6)	GNTK(8)	Ours(8)
PTC	62.9 ± 7.2	64.9 ± 5.3	63.5 ± 6.8	**66.2** ± **5.1**	65.2 ± 7.9	63.4 ± 6.6
NCI1	83.6 ± 2.1	**84.1** ± **1.2**	84.0 ± 0.9	83.8 ± 1.2	82.9 ± 1.8	82.3 ± 2.2