



Hybrid quantum neural networks for efficient protein-ligand binding affinity prediction

Seon-Geun Jeong¹, Kyeong-Hwan Moon² and Won-Joo Hwang^{1,2*}

*Correspondence:

wjhwang@pusan.ac.kr

¹ Department of Information Convergence Engineering, Pusan National University, Busandaehak-ro 63beon-gil, Geumjeong-street, Busan, 46241, Republic of Korea

² School of Computer Science and Engineering, Pusan National University, Busandaehak-ro 63beon-gil, Geumjeong-street, Busan, 46241, Republic of Korea

Abstract

Protein-ligand binding affinity is critical in drug discovery, but experimentally determining it is time-consuming and expensive. Artificial intelligence (AI) has been used to predict binding affinity, significantly accelerating this process. However, the high-performance requirements and vast datasets involved in affinity prediction demand increasingly large AI models, requiring substantial computational resources and training time. Quantum machine learning has emerged as a promising solution to these challenges. In particular, hybrid quantum-classical models can reduce the number of parameters while maintaining or improving performance compared to classical counterparts. Despite these advantages, challenges persist: why hybrid quantum models achieve these benefits, whether quantum neural networks (QNNs) can replace classical neural networks, and whether such models are feasible on noisy intermediate-scale quantum (NISQ) devices. This study addresses these challenges by proposing a hybrid quantum neural network (HQNN) that empirically demonstrates the capability to approximate non-linear functions in the latent feature space derived from classical embedding. The primary goal of this study is to achieve a parameter-efficient model in binding affinity prediction while ensuring feasibility on NISQ devices. Numerical results indicate that HQNN achieves comparable or superior performance and parameter efficiency compared to classical neural networks, underscoring its potential as a viable replacement. This study highlights the potential of hybrid QML in computational drug discovery, offering insights into its applicability and advantages in addressing the computational challenges of protein-ligand binding affinity prediction.

Keywords: Hybrid quantum neural network; Protein-ligand binding affinity; Quantum machine learning; Quantum neural network

1 Introduction

The effectiveness of a drug in interacting with a specific protein target is linked to the drug-target affinity, which is primarily determined by the structures of both the chemical and the protein [1]. Binding of a molecule to a protein may start a biological process. This includes the activation or inhibition of an enzyme's activity as well as the interaction be-

© The Author(s) 2025. **Open Access** This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

tween a drug molecule and its intended protein target. The binding is quantified by how strong the chemical compound. This metric quantifies how firmly the ligand, which is another term for the chemical compound, connects to the protein. Traditionally, dissociation (K_d), half inhibition concentrations (IC_{50}), and inhibition (K_i) constants have been utilized to represent experimentally determined binding affinities [2]. In drug discovery, a crucial selection criterion is the high binding affinity between a small molecule or short peptide and a receptor protein. Although the binding affinity could be measured directly through experimental methods, the time cost and financial expenses are extremely high due to insufficient known structures of protein-ligand complexes [1, 3]. Therefore, protein-ligand binding affinity prediction can serve significant advantage in the drug discovery.

In general, predicting the binding affinity can be categorized as: physics-based methods such as molecular docking [4], molecular dynamics simulations [5], have been widely used in binding affinity prediction. Similarity-based [6] or matrix factorization-based methods [7] predict binding affinity by utilizing the global similarity matrices of entire proteins or ligands. However, these traditional structure-based methods still remain challenging to identifying the binding ligand from a large-scale chemical space through currently experimental methods. In recent years, artificial intelligence (AI) and machine learning (ML) models have emerged as a promising tool for binding affinity prediction. Leveraging large-scale protein-ligand datasets, these models have demonstrated superior accuracy by learning complex relationships between molecular structures and binding strength [8]. For example, there are the deep learning methods for affinity prediction, such as Pafnucy [8], DeepAtom [9], TopologyNet [10], DeepDTA [11], WideDTA [12], and DeepDTAF [13]. These models not only outperform traditional methods in prediction accuracy but also exhibit high-potential of drug discovery based on ML. Despite these advancements, challenges still remain. One major issue is the increasing size and complexity of ML models, which result in higher computational costs and longer training times [14].

Quantum machine learning (QML) combines the computational principles of quantum mechanics, such as superposition and entanglement, with ML algorithms, offering significant advantages in computation speed [15–17]. Recently, the universal approximation property (UAP) of quantum neural networks (QNNs) has been investigated, which is similar to the universal approximation theorem (UAT) [18] in AI theory. In particular, a multi-qubit QNN model defined a partial Fourier series can be a universal function approximator [19]. Although the expressivity of QNNs is strong, there is limitation of expressivity of QNNs even though the QNN is made sufficiently deep [20].

One notable direction in QML research is the development of hybrid architectures that integrate classical ML and quantum computing. These hybrid models have demonstrated several advantages over purely classical counterparts [21]. For instance, they often require fewer parameters, leading to faster computations while maintaining or even enhancing performance. This highlights the potential for QML to eventually replace classical ML in drug discovery, offering a transformative leap in efficiency and capability [22]. However, hybrid quantum models are introduced but it remains challenges that the powerful expressivity comes from the classical part or the quantum part of hybrid models. Moreover, a systematic analysis of how parameters in the QNN affect the classes of functions that it can approximate is missing [23].

Furthermore, it is essential to evaluate the feasibility of implementing a quantum model on noisy intermediate-scale quantum (NISQ) devices. NISQ devices face significant con-

straints, including a restricted number of quantum qubits, vulnerability to quantum computational errors, and short coherence times, which pose challenges to their implementation [24, 25]. For example, the amplitude encoding method [26] requires deep circuit depth $O(\text{poly}(N))$, which may be infeasible due to the short coherence time of NISQ device. In contrast, angle embedding maintains a constant circuit depth but requires $O(N)$ qubits [27]. It requires a huge amount of qubits in large-dimensional input classical data. Therefore, designing a feasible encoding scheme on NISQ devices is crucial.

Contribution This study investigates and tackles several important challenges: 1) High computational costs and longer training times of classical ML models. 2) The potential of QNN in place of classical neural networks (NNs). 3) The feasibility of the proposed hybrid quantum model on NISQ devices. To address these challenges, we propose a novel QML-based method which is hybrid quantum DeepDTAF (HQDeepDTAF) to predict the protein-ligand binding affinity. In particular, to address the limitation of expressivity of QNN, the proposed model consists of a hybrid quantum model. To substitute the NN into hybrid quantum neural networks (HQNNs), we introduce data re-uploading models under the hybrid quantum model. For the target task, our structure is inspired by the DeepDTAF architecture, which consists of three separate modules: the entire protein module, the local pocket module, and the ligand simplified molecular input line entry system (SMILES) module. We follow the original module but the NN is substituted as a hybrid quantum model to achieve a parameter-efficient model in binding affinity prediction. Finally, we discuss the efficiency and feasibility of the proposed model. The main contributions of this study are summarized as follows:

- We propose a novel HQDeepDTAF for protein-ligand binding affinity prediction. Specifically, we introduce the hybrid embedding scheme to reduce the required qubit counts, and utilize classical regression network for prediction task.
- To address the limitation of expressivity of QNN, we propose a hybrid quantum framework for the target task. Specifically, our framework experimentally explores the various hybrid embedding schemes to find a suitable combination of quantum and classical. Moreover, to design the HQDeepDTAF model, we investigate the appropriate number of qubits and layers based on two key metrics: expressibility and entangling capability.
- We evaluated the effectiveness of our proposed algorithm by conducting a comparative analysis with state-of-the-art benchmarks using the protein data bank bind (PDBbind) dataset [28]. This assessment aimed to showcase the performance of our method in relation to existing state-of-the-art approaches.
- We performed a noise simulation to evaluate the effectiveness of our proposed algorithm on NISQ devices. Furthermore, we discuss the efficiency and feasibility of the proposed model.

2 Related works

This section presents the related work aligned with our quantum algorithm. The related work includes the universal approximation of QNN, as well as classical and quantum models for protein-ligand binding affinity prediction.

2.1 Universal approximation of quantum neural network

The UAT establishes that deep NNs can approximate well-behaved functions with arbitrary accuracy, forming the basis of their expressive power. QNNs, as quantum analogues of classical NNs, have been similarly studied for their expressivity under the UAT framework.

Authors in [29–31] have investigated the universal approximation of QNNs in terms of quantum activation functions or quantum neurons. They implemented the sigmoid and rectifier linear unit (ReLU) functions based on quantum neurons, which demonstrates that the quantum model can be used as a universal approximator. However, they required additional qubits, such as ancilla qubits, and multi-controlled gates for encoding the data into qubits. [23] has demonstrated that single-qubit QNNs can approximate any univariate function by mapping the model to a partial Fourier series. While the authors investigated the expressive power of a single-qubit QNN, this single-qubit QNN has a limited expressivity for multivariate functions. [20] demonstrated that sufficiently deep QNNs can approximate the target functions. While they exhibited that the loss is significantly reduced and approaches to zero, for adding nonlinearity in QNN, the quantum model requires additional ancillary qubits, which leads to serious computational resources. [19] has investigated the effect of data encoding on the expressivity of QNNs as function approximators. While the authors demonstrated that the multi-qubit QNNs have universality for multivariate functions, the quantum model requires exponential circuit depth, which is impractical to implement.

Authors in [32] proposed variational QSplines and generalized QSplines to approximate non-linear activation functions in QNNs. While both methods achieve good approximation performance for specific functions like sigmoid, ReLU, and ELU, they are not general-purpose and require tailored formulations per activation function. In addition, these methods use amplitude encoding, which provides logarithmic qubit scaling with respect to input size but induces polynomially increasing circuit depth as data dimension grows. They also require additional ancilla qubits for inner product computation and spline evaluation, increasing hardware overhead on NISQ devices. [33] proposed a variational quantum computing framework for solving nonlinear differential equations, notably the nonlinear Schrödinger equation. Their approach introduces a quantum nonlinear processing unit which combines multiple copies of variational quantum states to handle nonlinearities, along with tensor network-inspired quantum circuits to implement efficient operators. While their method demonstrates the feasibility of quantum approaches for nonlinear problems, it relies on constructing a quantum circuit with polynomially increasing depth on qubits and data dimension, with precise gate structures and encoding of nonlinear terms via specially crafted quantum circuits with an ancilla qubit.

Unlike existing studies that focus solely on pure QNNs based on the UAT and require additional qubits and exponential circuit depth, we propose a parameter-efficient hybrid quantum model designed for practical implementation on NISQ devices. While we do not claim theoretical universality, we empirically demonstrate the capability of our HQNN to approximate nonlinear functions across multiple benchmark tasks. To evaluate its potential as a substitute for classical NNs, we conduct comparative experiments between HQNN, pure QNNs, and classical NNs using both univariate and multivariate function benchmarks. Furthermore, we examine the expressivity of HQDeepDTAE, which integrates HQNN, in the context of protein–ligand binding affinity prediction. The feasibility

of the proposed model is assessed in terms of key NISQ constraints, including circuit depth and qubit count.

2.2 Protein-ligand binding affinity prediction

Leveraging large-scale protein-ligand datasets, ML models were developed for binding affinity prediction tasks. These models laid the foundation for modern advancements in drug discovery by utilizing protein-ligand interaction data to predict binding affinities with high accuracy.

Early approaches, such as Pafnucy [8], focused on deep learning-based regression techniques, incorporating features derived from protein-ligand structures to evaluate affinity. These efforts significantly improved the ability to predict drug-target interactions and were benchmarked rigorously against experimental datasets. Further research was conducted, such as models like DeepDTA [34], DeepDTAF [13], DeepAtom [35], DeepAffinity [36], and WidtDTA [12] which integrated convolutional neural networks (CNNs) for sequence-based feature extraction. These models emphasized the importance of representing both protein and ligand sequences effectively while demonstrating improved predictive capabilities over traditional docking methods. Additionally, TopologyNet [10] introduced a novel topology-based approach incorporating multi-task training to predict biomolecular properties alongside binding affinities. To address the limitations of static representations, Zeng et al. [11] employed multiple attention blocks to capture complex interaction patterns between ligands and binding sites. This attention mechanism not only enhanced the interpretability of predictions but also allowed the models to focus on critical interaction regions within the binding pockets. Similarly, hybrid frameworks, such as those described in Mohammad et al.'s research [9], incorporated structural data and sequence-based information to refine affinity predictions further.

For the QML-based protein-binding affinity prediction, Domingo, Laia, et al. [37] proposed the hybrid quantum-classical 3D CNN. In particular, the flexible representation of quantum image (FRQI) method was used to encode the image data into quantum states. They reduced 20% complexity than the classical counterparts while still maintaining optimal performance in the predictions. However, the FRQI method requires many quantum complex circuits for a single data sample. They showed the image in $(4 \times 4 \times 4)$ blocks, 32,832 quantum circuits are required. The current NISQ devices have limitations in the available number of qubits and quantum gates as well as short decoherence time. Dong, Lina, et al. [38] have shown that the quantum algorithm can achieve considerable accuracy, although the parameters used in the model have been remarkably reduced. In particular, quantum graph isomorphic networks, quantum graph convolution networks, and quantum CNNs have been constructed. They showed the potential of the hybrid quantum deep learning algorithm in bioinformatics. However, they left the noise effect on the proposed models in the NISQ devices. Domingo, L., et al. [39] substituted the end of the fully connected (FC) layer of both 3D CNN and spatial-graph CNN into a quantum fusion model. The study assessed the performance of the proposed model through a comprehensive comparison with a classical fusion benchmark. While the quantum fusion models outperformed their classical counterparts in both parameter efficiency and accuracy, the study did not address the practical feasibility of the quantum model or the limitations inherent to NISQ-era quantum devices.

Despite their critical impact on QML performance, previous studies have overlooked key factors such as the feasibility of the quantum model and NISQ device constraints such

as qubit counts, decoherence, and noise. To address this gap, we propose a novel hybrid quantum model within the practical limits of NISQ-era quantum computing. Based on this approach, we address the major issue of classical ML models, which is the complexity of the model in terms of the number of parameters in this study.

2.3 Hybrid quantum machine learning

Numerous studies have proposed a hybrid QML to leverage the advantage of quantum computing and classical ML algorithms in various domains.

[16, 17, 40] proposed a hybrid quantum-classical model for classification and prediction, demonstrating performance comparable to a classical model. They employed a combined FC layer with angle embedding or amplitude without a data re-uploading scheme for quantum data encoding. In addition, these studies did not consider noise effects in NISQ devices. [41, 42] introduced multiple parallel quantum circuits with angle embedding for image classification. This approach showed a reduction of computation time in terms of parameter count. The model outperformed a classical CNN or QCNN baseline on the MNIST dataset in accuracy. However, they didn't analyze noise effects experiments for multiple parallel quantum circuits for quantum features in NISQ devices. [43] proposed an actor-critic-based quantum deep reinforcement learning model for collision-free navigation in self-driving cars, employing pure angle embedding in the critic. However, this embedding scheme is limited in handling high-dimensional inputs. While the proposed model demonstrated improved training stability and slightly higher average cumulative rewards compared to its classical counterpart, its performance significantly deteriorated under noise simulations.

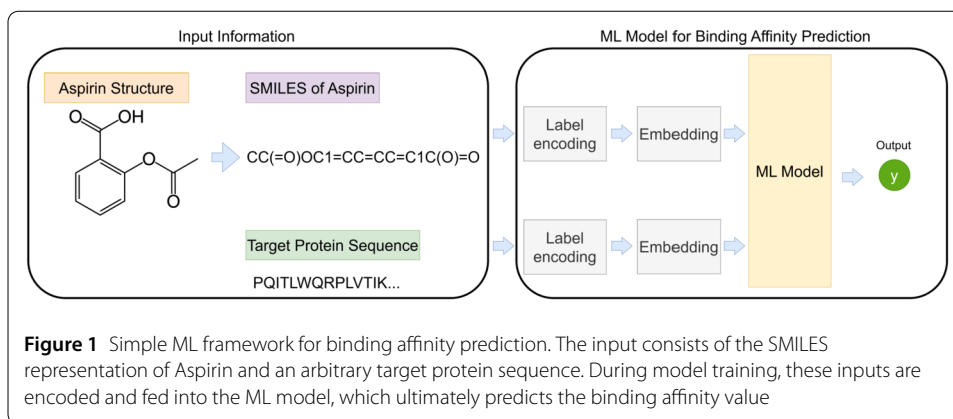
While previous studies focus on pure quantum embeddings, FC-based hybrid embeddings, or multiple parallel quantum circuits, our approach adopts a hybrid architecture that differs in its embedding strategy. Specifically, we employ a data re-uploading scheme that integrates a classical embedding network with quantum angle embedding. Unlike prior work, our classical embedding can flexibly incorporate a wide range of encoders, such as NNs, CNNs, and other general-purpose architectures, enabling broader applicability and adaptability. This design supports flexible qubit allocation and enhances QNN performance. The proposed HQDeepDTAF is carefully tailored to NISQ constraints, offering a parameter-efficient architecture that ensures both performance and hardware feasibility.

3 Background & preliminaries

In this section, we present the necessary background to understand the proposed algorithm. We begin by describing the binding affinity prediction problem. Then, we introduce key concepts in QML, including quantum states, quantum encoding, and data re-uploading. Finally, we discuss the UAT, which forms the theoretical foundation of our approach.

3.1 Machine learning-based binding affinity prediction

Binding affinity refers to the strength of the interaction between a drug and its receptor, and it is a key property for understanding how a drug's structure influences its function [44]. Binding affinity prediction involves estimating the interaction strength between a ligand (e.g., a drug molecule) and its target protein. Traditionally, this has been measured through experimental methods, which are accurate but often time-consuming and



costly [2]. To overcome these limitations, ML models have emerged as efficient alternatives [8]. ML-based binding affinity prediction typically relies on experimentally derived datasets, where binding affinities are expressed in terms of constants such as K_d , IC_{50} , and K_i . The objective of these models is to predict the binding strength based on molecular and protein information. Input representations can vary but commonly include the structures or sequences of proteins and ligands, which are often transformed into SMILES format. As illustrated in Fig. 1, we present a simple ML framework for binding affinity prediction. In this example, the input consists of the SMILES representation of Aspirin and an arbitrary target protein sequence. These inputs are encoded and fed into the ML model, which is trained to predict the resulting binding affinity value.

3.2 Quantum state

The basic unit of information in quantum computation is one quantum bit, or qubit for short. A classical bit has a state based on Shannon information, with the state being either 0 or 1. However, the qubit can exist in a superposition of both the 1 and 0 states, the state of a single qubit is a unit vector in a 2-dimensional Hilbert space $\mathbb{C}^2$, which is commonly denoted in Dirac notation as follows:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

where α and β are complex numbers with $|\alpha|^2 + |\beta|^2 = 1$. $|0\rangle = (1, 0)^T$ and $|1\rangle = (0, 1)^T$ are known as computational basis states. When qubits are measured, the result is always either 0 or 1; the probabilities of these outcomes depend on the quantum state of the qubits before the measurement. In addition, a quantum state of n qubits can be represented as a normalized vector in the n -fold tensor product Hilbert space $\mathbb{C}^{2^n}$.

3.3 Quantum encoding

Quantum encoding strategies, involving conversion classical data into quantum states, affects to the performance of quantum model directly [45]. Consequently, it is crucial to employ appropriate quantum embedding techniques to encode classical data into a quantum system. Numerous quantum embedding methods have been proposed [26, 27], where angle embedding and amplitude embedding are widely used. An angle embedding method transforms the classical information into the angle of rotation $\theta \in [0, \pi]$. For example,

given a input data $\mathbf{x} = [x_1, \dots, x_n]^T \in \mathbb{R}^N$, the angle embedding method maps all information into $O(N)$ qubits with a constant-depth quantum circuit as follows:

$$U_\phi : \mathbf{x} \rightarrow |\phi(\mathbf{x})\rangle = \bigotimes_{i=1}^N \left(\cos\left(\frac{x_i}{2}\right) |0\rangle + \sin\left(\frac{x_i}{2}\right) |1\rangle \right), \quad (1)$$

where $x_i \in [0, \pi)$ for all i . The angle embedding approach necessitates numerous qubits when dealing with substantial amounts of information. Thus, it is difficult to all information encodes to quantum states without loss of information due to the capability of current NISQ devices. In contrast, the amplitude embedding method has the advantage of reducing the number of qubits $O(\lceil \log(N) \rceil)$ for input data $\mathbf{x} \in \mathbb{R}^N$. Therefore, it can represent large amounts of information using a small number of qubits. For instance, for the normalized input data $\bar{\mathbf{x}} \in [0, 1]^N$, each data $\bar{x}_i$ can be encoded into $\lceil \log(N) \rceil$ qubits as follows:

$$U_\phi : \bar{\mathbf{x}} \rightarrow |\phi(\bar{\mathbf{x}})\rangle = \sum_{i=1}^{2^{\lceil \log(N) \rceil}} \bar{x}_i |i\rangle, \quad (2)$$

where $\sum_i |\bar{x}_i|^2 = 1$. Although the amplitude embedding method requires a smaller number of qubits than the angle embedding method, the amplitude embedding has a circuit depth of $O(\text{poly}(N))$ [26]. Therefore, the embedding method should be selected while considering the NISQ device capability and model performance.

3.4 Data re-uploading

The data re-uploading QNN model [18] is a generalized framework of QML models based on parameterized quantum circuits (PQCs). Given a set of input data $\mathbf{x} = \{x_1, \dots, x_n\} \in \mathbb{R}^N$ and a set of trainable parameters $\boldsymbol{\theta} = \{\theta_0, \dots, \theta_1\}$, a data re-uploading QNN is a quantum circuit that consists of interleaved data encoding circuit blocks $S(\cdot)$ and PQC blocks $P(\cdot)$ as follows:

$$U_{\theta,L} = P(\theta_0) \prod_{i=1}^L S(\mathbf{x}) P(\theta_i), \quad (3)$$

where L denotes the number of PQC block layers. For the N qubit system, the output of data re-uploading QNN model with observables $M_\beta(Z^{\beta_1} \otimes Z^{\beta_2} \otimes \dots \otimes Z^{\beta_n})$ can be represented as follows:

$$f_{\theta,L} = \langle 0|^{\otimes N} U_{\theta,L}^\dagger M_\beta U_{\theta,L} |0\rangle^{\otimes N}. \quad (4)$$

3.5 Universal approximation theorem

The UAT plays an essential role in the development of NNs, which states that sufficiently wide or sufficiently deep defined NNs can approximate an arbitrary function with arbitrary accuracy [46]. This theorem lays the foundation of the expressive capability of NNs and serves as a basis for the success of NN applications.

Theorem 1 (Universal Approximation Theorem of classical NN [18]) *For any Lebesgue integer function $f : \mathbb{R}^N \rightarrow \mathbb{R}$ and input data $\mathbf{x} = \{x_1, \dots, x_n\} \in \mathbb{R}^N$, there exists a FC classical*

NN with function $\varphi : \mathbb{R} \rightarrow \mathbb{R}$, the NN $F(\mathbf{x}) = \sum_{i=1}^N \alpha_i \varphi(w_i x_i + b_i)$ with $\alpha_i, b_i \in \mathbb{R}$ and $w_i \in \mathbb{R}^N$ such that F with any precision $\varepsilon > 0$ satisfies

$$|f(\mathbf{x}) - F(\mathbf{x})| dx < \varepsilon, \quad (5)$$

where a Lebesgue-integral function $f : \mathbb{R}^N \rightarrow \mathbb{R}$ is a Lebesgue-measurable function satisfying $\int_{\mathbb{R}^N} |f(x)| dx < \infty$ which contains continuous functions. In classical NNs, φ denotes the activation function, w are the weights for each neuron, b are the biases and α are the neuron weights that construct the output function. Therefore, this theorem establishes that it is possible to reconstruct any continuous function with a single layer NN of N neurons. The proof of this theorem for ReLU activation function $\text{ReLU}(x) = \max\{x, 0\}$ can be found in [46].

4 Methodology

4.1 Main architecture

The proposed HQDeepDTAF is shown in Fig. 2, consists of four main modules: 1) Input information; 2) Embedding layer; 3) Classical convolution module; 4) HQNNs for protein-ligand affinity prediction. For the input information, ligand representation, protein representation, and pocket representation are used as input data. These representations have proved to be beneficial for affinity prediction [13]. Subsequently, input information is used for three different classical convolution modules for important feature extraction. The outcome of each module is concatenated to learn their interactions and predict the protein-ligand binding affinity via the HQNN. In the following, we first describe the input information and then describe classical convolution modules, and HQNN step by step.

4.1.1 Input information

As input data, we employed representations of the ligand, protein, and pocket, which can contribute to improved affinity prediction [13]. Note that the input information of the model is 1-dimensional sequence data because the 3-dimensional structures of some proteins are still unknown. The details of the input information are listed as follows.

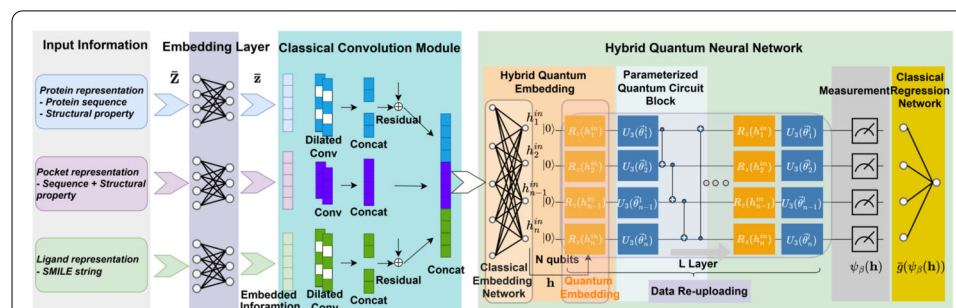


Figure 2 Schematic of the HQDeepDTAF model. The representations of the protein, pocket, and ligand are provided as inputs. These inputs are first processed by an embedding layer. The classical convolution module consists of dedicated convolutional layers for each input type. The outputs from the three modules are then concatenated and passed to the HQNN. The hybrid quantum embedding, which combines a classical embedding network with a quantum embedding, encodes the data into a data re-uploading QNN. The measurement outcomes of the QNN are then passed to a classical regression network, which predicts the protein-ligand binding affinity

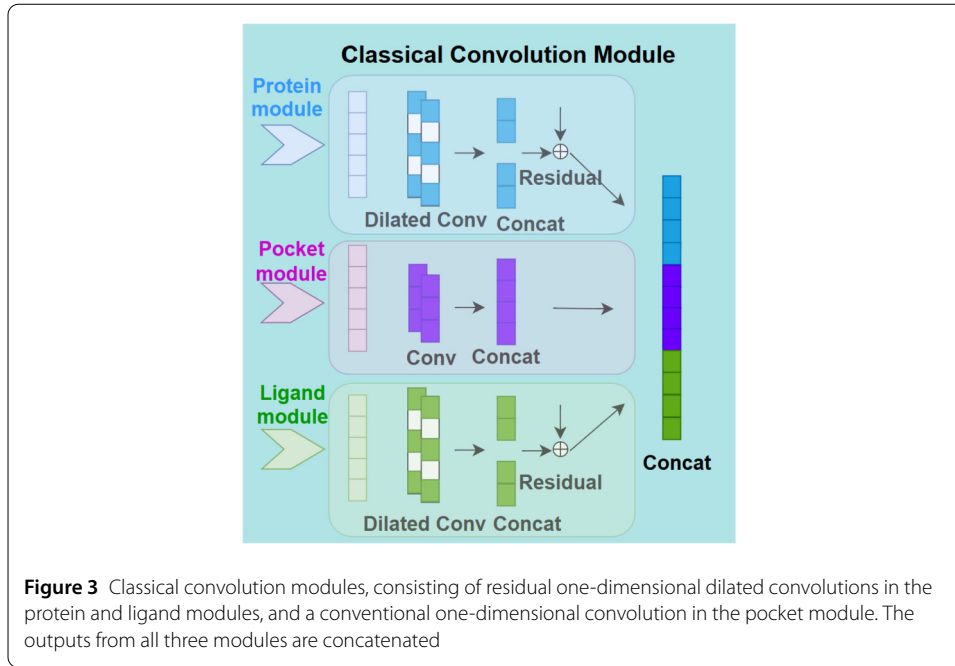
Ligand representation The widely used 1-dimensional representation for ligand chemical structures is the SMILES [47], which encodes molecular structures as character sequences that denote atoms, bonds, rings, and connectivity patterns. In this study, all ligand structure data format (SDF) files were converted into SMILES strings using Open Babel [48] to ensure a consistent molecular representation. Each SMILES string was subsequently mapped to a fixed-length numerical encoding scheme based on a predefined vocabulary of 64 distinct characters. Each character was assigned a unique integer identifier, such as 'C' represented as 42, 'O' as 48, '=' as 40, ')' as 31, and '(' as 1. For example, the SMILES string "CC(=O)CC" was transformed into the sequence "42, 42, 1, 40, 48, 31, 42, 42".

Protein representation In this study, as global features, protein representation consists of sequence representation and structural property representation. These integrating sequence and structural information leads to more accurate and reliable prediction than using either representation alone [49]. Regarding protein sequence representation, a basic approach involves depicting the molecular structure as a one-dimensional sequence using a 20-amino acid alphabet. Here, we utilized a one-hot vector consisting of 21 dimensions to encode the 21 various residue types present in protein sequences. Note that we opted for a 21-dimensional vector instead of a 20-dimensional one to accommodate non-standard residues present in certain proteins. The structural property representation encompassed both secondary structure elements (SSEs) [50] and physicochemical attributes [13]. To predict secondary structure for each sequence, the SSPro program [51] was employed. Following this, SSEs were encoded using an 8-dimensional one-hot vector. Physicochemical characteristics were encoded by an 11-dimensional vector. Consequently, a 19-dimensional vector was utilized to depict the structural property of each residue. In total, a 40-dimensional feature vector was employed for each residue to characterize global protein features, combining both sequence and structural property representations.

Pocket representation The pocket refers to a binding cavity in a protein, defined by specific physicochemical properties, shape, and location, which determine protein function. Protein-ligand interactions primarily depend on ligand binding to these pockets, which are composed of key amino acids from discontinuous sequences [52]. Therefore, a pocket representation is considered comprehensive for extracting local features. In protein-ligand binding affinity prediction, these local pocket features play a crucial role and are utilized as the primary input information. To encode local pocket features, a 40-dimensional feature vector was employed for each pocket residue. This vector combines sequence representation and structural property representation, as outlined in the protein representation.

4.1.2 Embedding layer

We used an embedding layer to represent inputs with dense vectors in three modules. The embedding layer converts sparse, 1-dimensional sequence data inputs into dense vectors, making features more suitable for the model. In models where distinct features are processed through separate embedding layers, the layers are not shared because each feature represents fundamentally different information types. Separate embedding layers allow the model to learn feature-specific representations without interference.



Definition 1 (Embedding Layer) Given the input data $\bar{\mathbf{Z}} = [\bar{Z}_1, \dots, \bar{Z}_n]$, embedding layer transform input features into K -dimensional dense vectors as follows:

$$L_e : \bar{\mathbf{Z}} \rightarrow \bar{\mathbf{z}} = [\bar{z}_1, \dots, \bar{z}_k], \quad (6)$$

where the k -th component of the vector is given by $\sigma(w_{k1}\bar{Z}_1 + \dots + w_{kn}\bar{Z}_n)$, with $\sigma(\cdot)$ denoting the activation function and w representing the trainable weight parameters.

4.1.3 Classical convolution modules

The embedded information obtained from Definition 1 is used to three different convolution modules as shown in Fig. 3. With respect to the protein and ligand module, the dilated convolution [53] and residual learning [54] are introduced. For pocket module, we used vanilla convolution. For the design of classical convolution modules, we follows the Deep-DTAF design [13]. In the protein module, a one-dimensional dilated convolution with five different dilation rates was employed to account for long-range interactions in extended protein sequences. These dilated convolution layers were followed by a max pooling layer, similar to the ligand module. However, in the ligand module, the dilated convolution utilized four different dilation rates. For the pocket module, three one-dimensional conventional convolution layers were applied, progressively increasing the number of filters, followed by a max pooling layer. Finally, the outputs from the max pooling layers of all three modules were concatenated and passed into the HQNN component.

Definition 2 (Dilated Convolution [53]) Given $F : \mathbb{Z}^2 \rightarrow R$ is discrete function, $\Omega_r = [-r, r]^2 \cap \mathbb{Z}^2$, and $k : \Omega_r \rightarrow R$ is a discrete filter of size $(2r + 1)^2$. The dilated convolution

operator $*_l$ with dilation rate l is defined as follows:

$$(F *_l k)(p) = \sum_{s+lt=p} F(s)k(t), \quad (7)$$

where s and t are subscripts of element vectors.

To capture extensive interactions between protein features and ligand SMILES, Definition 2 is employed, which expanded the effective size of the receptive field.

4.1.4 Hybrid quantum neural network

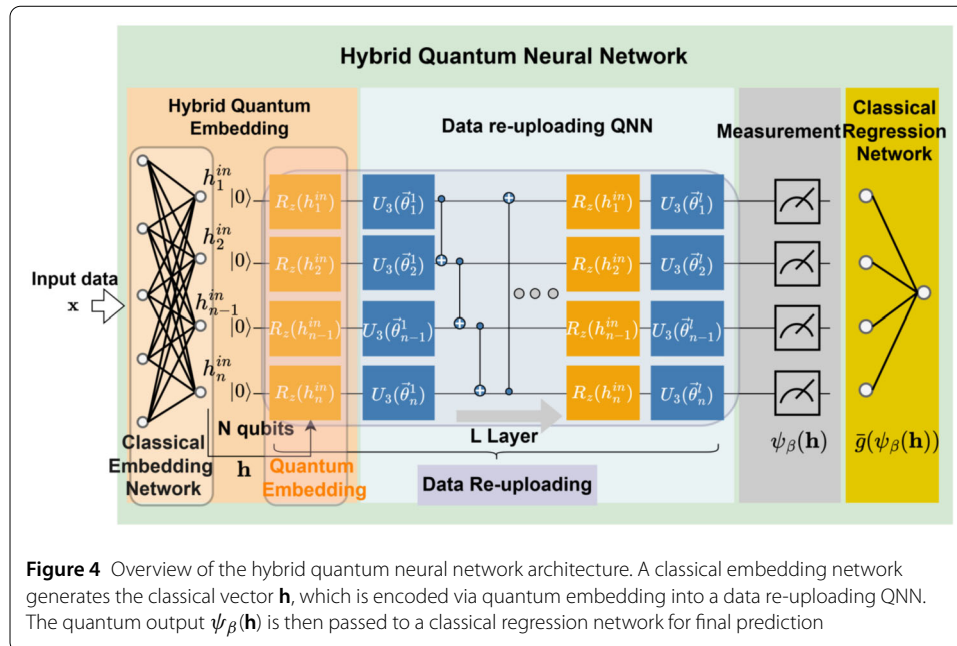
As shown in Fig. 4, the proposed HQNN comprises a classical embedding network, a data re-uploading QNN, and a classical regression network. The hybrid quantum embedding, combining a classical embedding network with quantum embedding, allows flexible control over the number of qubits, which is essential for implementation on NISQ devices with limited qubit counts. Subsequently, the quantum computation for quantum states is conducted by the PQC blocks with data re-uploading scheme.

We use a classical embedding network, which can be NN, CNN, and various types of NN, as the embedding module to project input data $\mathbf{x} = \{x_1, \dots, x_d\} \in \mathbb{R}^D$ into a lower-dimensional latent space $\mathbf{h} = [h_1^{in}, \dots, h_n^{in}]^T$, where each component is given by:

$$h_n^{in} = \sigma(\omega_{n1}x_1 + \dots + \omega_{nd}x_d), \quad (8)$$

with weights ω and activation function $\sigma(\cdot)$. This latent vector $\mathbf{h}$ is then mapped to a Hilbert space using a quantum embedding U_ϕ , as formalized below.

Definition 3 (Hybrid Quantum Embedding) Let $\mathcal{H}$ be a Hilbert space, and let $\mathbf{h}$ be the output of a classical embedding network. A quantum embedding circuit U_ϕ maps $\mathbf{h} \mapsto |\phi(\mathbf{h})\rangle \in \mathcal{H}$, where $|\phi(\mathbf{h})\rangle$ is the encoded feature vector.



Example 1 (NN-Angle Embedding) Using angle embedding, each $h_i^{in} \in \mathbf{h} \in \mathbb{R}^N$ is mapped as:

$$U_\phi(\mathbf{h}) = \bigotimes_{i=1}^N \left(\cos\left(\frac{h_i^{in}}{2}\right) |0\rangle + \sin\left(\frac{h_i^{in}}{2}\right) |1\rangle \right). \quad (9)$$

We adopt a data re-uploading QNN architecture, in which the quantum circuit alternates between PQC blocks and data re-uploading blocks defined as

$$S(\mathbf{h})_H := e^{-ih_1^{in}H} \otimes \dots \otimes e^{-ih_n^{in}H}, \quad (10)$$

where $H \in \{\sigma_x, \sigma_y, \sigma_z\}$ is a fixed Pauli operator. For simplicity, we employ angle embedding to encode classical data into quantum rotations.

The full QNN unitary with L re-uploading layers is expressed as:

$$U(\mathbf{h}, \theta) = P^{(L+1)} S(\mathbf{h})_H P^{(L)} \dots S(\mathbf{h})_H P^{(1)} = P(\theta_0) \prod_{i=1}^L S(\mathbf{h})_H P(\theta_i). \quad (11)$$

The PQC block $P(\theta)$ uses single-qubit rotations and CNOT entanglers inspired by [55]:

Definition 4 (Parameterized Quantum Circuit Block) Assume $\forall i, R_i \in \mathbf{SU}(2)$ are single-qubit rotations with three trainable parameters of Eq. (A.2). $\forall j, T_{t_j}$ is a single-qubit unitaries, then the PQC blocks P is defined as follows:

$$P = \prod_{j=0}^{n-1} E_{c_j}(T_{t_j}) \prod_{i=0}^{n-1} R_i, \quad (12)$$

where the single-qubit unitary is defined as Pauli X. Thus, the $E_{c_j}(T_{t_j}) \in \mathbf{U}(4)$ can be expressed as CNOT gate. In addition, $E_{c_j}(T_{t_j})$ are applied as a layer of $n/\gcd(n, r)$ controlled gates, where r is a hyperparameter called the range, $\gcd(n, r)$ is the greatest common divisor of given n and r , and $0 < r < n$. Specifically, For $j \in [1, \dots, n/\gcd(n, r)]$, the j -th 2-qubit gate $E_{c_j}(T_{t_j})$ of an PQC blocks have wire number $t_j = (jr - r) \bmod n$ as the control, wire number $c_j = jr \bmod n$ as target.

Each PQC block is hardware-compatible with NISQ constraints, consisting only of single-qubit rotations and CNOTs. As proven in Appendix A, any such unitary can be decomposed using the Z-Y decomposition [56].

Definition 5 (Data Re-uploading QNN with Classical Embedding Network) Given input $\mathbf{x} \in [0, 1]^D$, embedding output $\mathbf{h}$, and observable $O_\beta = Z^{\beta_1} \otimes \dots \otimes Z^{\beta_n}$, the QNN output is defined as:

$$\psi_\beta(\mathbf{h}) = \langle 0|^{\otimes N} U^\dagger(\mathbf{h}, \theta) O_\beta U(\mathbf{h}, \theta) |0\rangle^{\otimes N}. \quad (13)$$

Definition 5 formalizes the data re-uploading QNN with a classical embedding network, distinguishing it from HQNN, which additionally includes a classical regression network.

While increasing circuit depth improves QNN expressivity, pure QNNs still struggle to approximate general well-behaved functions due to inherent limitations [20]. One remedy is to increase the Hilbert space dimension by using more qubits, but this is impractical on NISQ devices due to limited qubit counts and coherence times. To overcome this, we incorporate a classical embedding network that both compresses input features and optimizes the quantum embedding during training, as supported by Theorem 1. This allows for efficient use of limited qubits while maintaining expressive capacity.

To further enhance performance under NISQ constraints, we extend the structure defined in Definition 5 by introducing a classical regression network, resulting in the HQNN architecture illustrated in Fig. 4. The definition of the HQNN model is as follows.

Definition 6 (Hybrid Quantum Neural Network) Let $\mathbf{x} = \{x_1, \dots, x_d\} \in [0, 1]^D$ be the input data and $N \geq D$ the number of qubits. Let $\mathbf{h}$ be the output of a classical embedding network, and $O_{\beta} = Z^{\beta_1} \otimes \dots \otimes Z^{\beta_n}$ the observable with $\beta \in \{0, 1\}^N$. The QNN output $\psi_{\beta}(\mathbf{h})$ follows Definition 5. We define a classical regression network $\bar{g}: \mathbb{R}^K \rightarrow \mathbb{R}$ applied to K -dimensional QNN outputs ($K \leq N$), where the final prediction is given by:

$$\bar{g}(\psi_{\beta}(\mathbf{h})) = \sum_{i=1}^K \sigma(w_i \psi_{\beta_i} + b_i), \quad (14)$$

with weights $w_i \in \mathbb{R}$ and biases $b_i \in \mathbb{R}$.

4.2 Parameter shift rules for quantum neural network training

A quantum circuit driven by our framework learns a given task by updating parameters using parameter shift rules [57]. Consider quantum states $|\psi\rangle$ comprising of N qubits, unitary U with trainable parameters θ , the expectation value with Pauli operators $B_j \in \{I, X, Y, Z\}^{\otimes n}$ can be represented as follows: $\langle \mathbf{B} \rangle = \langle \psi | U^{\dagger}(\theta) B_j U(\theta) | \psi \rangle$. Suppose expectation value is loss function $\mathcal{L}(\theta)$ of a quantum circuit, then with a gradient-based strategy, the update of the trainable parameters θ of QNN at the step t can be expressed as follows:

$$\theta_{t+1} = \theta_t - \eta \nabla \mathcal{L}(\theta_t), \quad (15)$$

where the gradient $\nabla \mathcal{L}(\theta_t) = \frac{1}{2} \langle \psi | U^{\dagger}(\theta_t + \frac{\pi}{2}) B_j U(\theta_t + \frac{\pi}{2}) | \psi \rangle - \frac{1}{2} \langle \psi | U^{\dagger}(\theta_t - \frac{\pi}{2}) B_j U(\theta_t - \frac{\pi}{2}) | \psi \rangle$. That is, the gradient can be calculated just by shifting $\pm\pi/2$ rotation on unitaries U .

5 Results

In this section, we provide a detailed description and analysis of the experiments conducted to evaluate the performance of the proposed HQDeepDTAF. The subsections provide a brief overview of the dataset, experimental settings, and experiments employed in the prediction task. In addition, our experiment was designed with two primary objectives. The first objective was to demonstrate the function approximation capabilities of HQNN, which was analyzed to be more efficient in function approximation than QNN and NN. Subsequently, HQDeepDTAF experiments were conducted for protein-ligand binding affinity prediction.

5.1 Experimental settings

1) Data description: PDBbind database [28] includes a collection of experimentally verified protein-ligand binding affinity expressed with $-\log(K_i)$, $-\log(K_d)$, $-\log(IC_{50})$ from the PDB [3]. For our study, we selected the core 2016 dataset from the PDBbind database version 2016, which is commonly utilized as a high-quality benchmark [13]. This dataset encompasses a wide range of structures and binding information, making it suitable for assessing various docking techniques. The provided datasets included protein PDB files, pocket PDB files, and ligand SDF files, among others. We extracted the protein and pocket sequences from the PDB files. Additionally, we transformed the SDF files into SMILES strings.

2) Data settings: To create an effective representation format, it is essential to establish fixed lengths for protein sequences, pocket sequences, and SMILES strings, as they vary in length. Based on [13], to match the same experimental setting as the counterpart model on overall datasets, predetermined lengths for sequences of proteins, pockets, and SMILES strings were selected. As a result, we established fixed character limits for different sequence types: 1000 characters for protein sequences, 150 characters for SMILES strings, and 63 characters for pocket sequences. These limits were chosen to encompass approximately 90% of the proteins, ligands, and pockets found in the datasets we analyzed. The sequences that are longer than the fixed characters were truncated and the sequences that are shorter than the fixed characters were 0 padded. The core 2016 set was compiled to test and evaluate our model. The Smith–Waterman similarity [60] for each protein sequence in the core 2016 test set was at most 60% [11] to any sequence in the training set for 99% of protein pairs. While training, we randomly load stochastic datasets for each iteration with random shuffling.

3) Baselines: In the experiments, to demonstrate the proposed HQDeepDTAF algorithm's effectiveness, we compared it with the state-of-the-art benchmarks, DeepDTAF [13], DeepDTA [34], Pafnucy [8], TopologyNet [10], AEScore [58], and K_{deep} [59]. To ensure a fair and reproducible evaluation, we implemented all classical and quantum models using consistent training configurations and dataset preprocessing. Both of DeepDTAF and HQDeepDTAF use the same structural classical convolution modules. In this study, we explored different embedding techniques to determine the most appropriate method for our proposed algorithm. Specifically, HQDeepDTAF-Amplitude incorporates a QNN based on amplitude embedding, coupled with a classical regression layer. In comparison, HQDeepDTAF-NN-Angle and HQDeepDTAF-NN-Amplitude adheres to Definition 6, utilizing a hybrid embedding approach as outlined in Definition 3. Note that, regarding HQDeepDTAF-NN-Amplitude model, at most 9-qubits is introduced to set the similar total number of parameters with DeepDTAF. For baselines, we followed the architecture and training procedures described [8, 10, 13, 34, 58, 59]. However, TopologyNet was reimplemented using only protein sequence inputs, as its original implementation is not publicly available. Table 1 provides a comprehensive overview of the structural specifications for these models used in our experiments. All implementations, including reimplemented versions of some baselines, are made publicly available on our GitHub repository <https://github.com/drmoon-1st/HQDeepDTAF>.

4) Training settings: In terms of convergence time, all models were trained for 20 epochs using a batch size of 16. The adaptive moment optimizer called AdamW [61] optimizer, which improves upon the original Adam by decoupling weight decay from the gradient

Table 1 Summary of Models used for binding affinity prediction

Model	Architecture details
DeepDTAF [13]	For all embedding network, output dimension was 128. For protein module, five 1D dilated convolutional layers with dilated rates of 1, 2, 4, 8, 16. For pocket module, three 1D convolutional layers with 32, 64, 128 filters. For ligand module, four 1D dilated convolutional layers with dilated rates of 1, 2, 4, 8. For all modules, convolutional filter size was 3 with max pooling layer after convolution. The outcomes of each module are concatenated and input to the dense layers. Dense consists of three layers with 128, 64, 1 nodes, each followed by dropout layer of rate 0.5.
DeepDTA [34]	The DeepDTA employs two CNN blocks followed by dense layer. Each CNN block consists of three 1D convolutions of 32, 64, 96 filters. Dense layer consists of 1024, 1024, and 512 neurons in sequence.
Pafnucy [8]	The Pafnucy architecture employs 3D convolutional layers with 64, 128, and 256 filters, each followed by max pooling. The output from the final convolutional layer is flattened and fed into dense layers consisting of 1000, 500, and 200 neurons in sequence.
TopologyNet [10]	https://github.com/drmoon-1st/HQDeepDTAF
AEScore [58]	https://github.com/abdulsalam-bande/KDeep
K_{deep} [59]	https://github.com/RMeli/aescore.git
HQDeepDTAF-Amplitude	For QNN with NN-based classical regression network, the concatenated results from each model are processed into 512D data using zero padding. The 9-qubits data re-uploading QNN followed.
HQDeepDTAF-NN-Amplitude	For HQNN with amplitude embedding, the NN-based classical embedding network with 512 nodes serves as input for a 9-qubit data re-uploading QNN consisting of 20 layers.
HQDeepDTAF-NN-Angle	For HQNN with angle embedding, the outcome of the NN-based classical embedding network with 9 nodes serves as input for 9-qubits data re-uploading QNN with 20 layers.

update. In contrast to traditional L2 regularization, AdamW applies weight decay directly to the weights, rather than through the gradients. The update rule is given by:

$$\theta_{t+1} = \theta_t - \eta \left(\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} + \lambda \theta_t \right), \quad (16)$$

where θ_t denotes the model parameters at step t , $\hat{m}_t$ and $\hat{v}_t$ are the bias-corrected first and second moment estimates, η is the learning rate, and λ is the weight decay coefficient. This decoupled formulation leads to improved generalization and more stable training. In our experiment, we use a max 0.005 learning rate, and the weight decay of 0.01 was utilized for training each model. Mean squared error (MSE) is used for the loss function for training our model. MSE can be defined as follows:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2, \quad (17)$$

where n denotes the number of samples, y_i for the target that can be also defined as $-\log(K_d/K_i)$, and $\hat{y}_i$ for the prediction value from our model. The process was repeated five times, and the average prediction value was computed using the trained models with the lowest training loss value during the training process. The resulting values were used

for comparison and discussion purposes. All experiments were conducted on a computer equipped with an AMD Ryzen 9 7950X 16-core Processor central processing unit (CPU) and 128 GB of RAM. We utilized PennyLane [62], a widely used tool, to implement the quantum model. On the other hand, we employed PyTorch to implement the classical models. Note that, due to the limitations of our environment, we used only a classical computer in this study.

5) Metrics: In assessing protein-ligand binding affinity prediction, we compared the predicted values with experimentally determined affinity measurements. To gauge our model's effectiveness, we employed mean absolute error (MAE), and root mean square error (RMSE) as indicators of prediction accuracy. We aimed to evaluate the correlation between predicted and experimental affinity values using the Pearson correlation coefficient (R) [63] and standard deviation (SD) [64] in regression analysis. The SD in regression was calculated as follows [13]:

$$SD = \sqrt{\frac{1}{N-1} \sum_{i=1}^N [y_i - (ap_i + b)]^2}.$$

Here, N represents the total number of protein-ligand complexes, while y_i and p_i denote the actual and predicted affinity for the i th complex, respectively. The variables a and b correspond to the slope and intercept of the linear function between actual and predicted values. Additionally, we utilized the concordance index (CI) [65], which represents the probability that the predicted and true affinity values for two randomly chosen protein-ligand complexes are in a specific order. The equation for CI is expressed as [13]:

$$CI = \frac{1}{Z} \sum_{y_i > y_j} h(p_i - p_j),$$

where p_i represents the predicted value corresponding to the higher binding affinity value y_i , while p_j denotes the predicted value for the smaller affinity value y_j . The variable Z serves as a normalization constant, representing the total count of protein-ligand complexes. The function $h(u)$ is defined as follows: it equals 1.0 when $u > 0$, 0.5 when $u = 0$, and 0.0 when $u < 0$. A stronger correlation between the model's predicted values and experimentally measured affinity values is indicated by a larger R value, lower SD, and higher CI value.

6) Experiments: We conducted a series of experiments to evaluate the performance of the proposed HQNN model. These include:

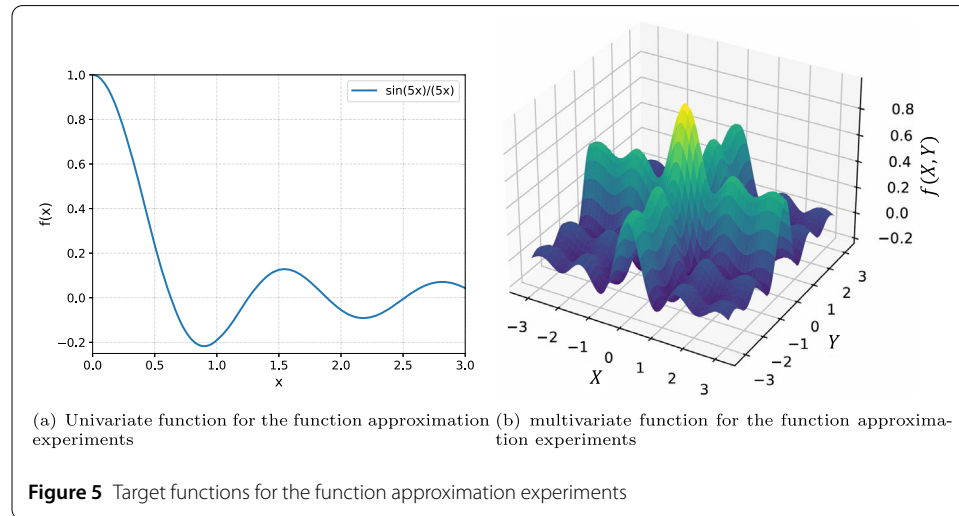
- analysis of function approximation of HQNN
- analysis of layer and qubit counts effects
- analysis of the effect of the encoding scheme
- ablation study on HQNN components
- analysis of binding affinity prediction
- analysis of convergence speed
- analysis of the noise effects

5.2 Analysis of function approximation of HQNN

We investigate that HQNNs can effectively serve as non-linear function approximators and replace classical NNs. To demonstrate this, we concentrate on regression problems, as

Table 2 Experimental Dataset for Performance Evaluation

Functions	Function type	Interval	Training	Test
$\sin(5x)/(5x)$	univariate function	$x \in (0, 3]$	200	100
$\sin(5x_1)/(5x_1) + \sin(5x_2)/(5x_2)$	multivariate function	$x_1, x_2 \in (0, 3]$	200	100

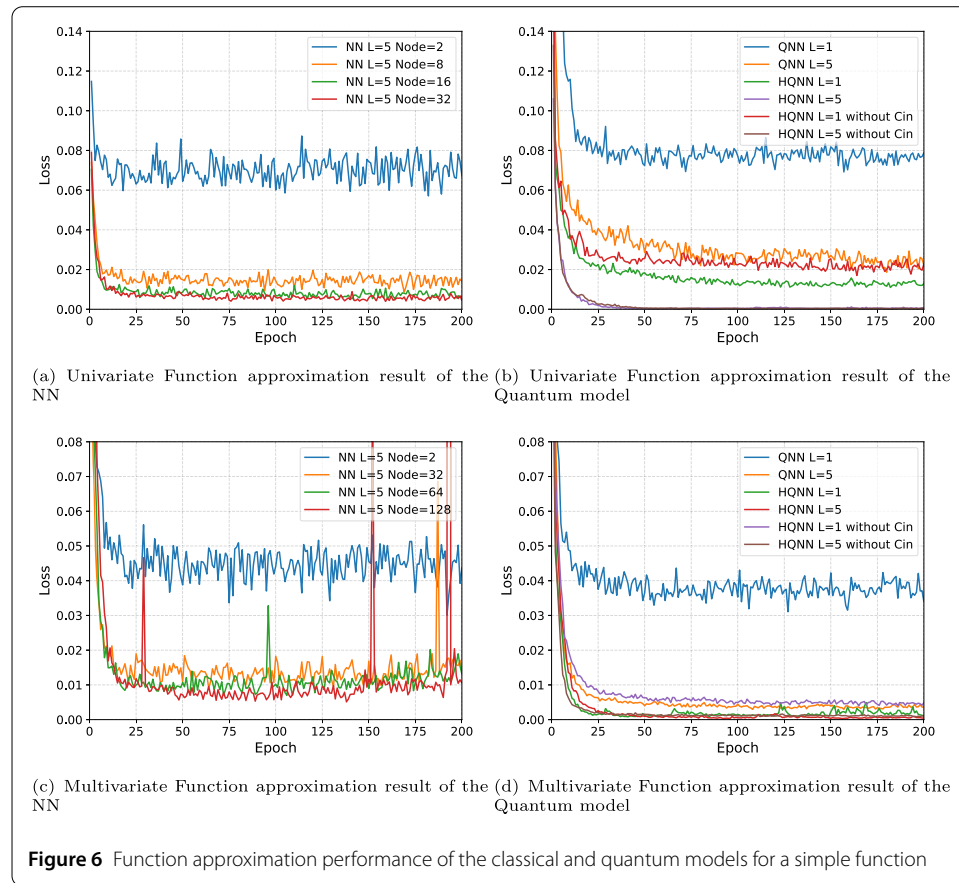


binding affinity prediction falls under this category of tasks. Herein, the experiments are widely used to demonstrate the capability of approximating non-linear functions of the model [19, 20, 23]. In our experimental setup, we evaluate both single-qubit and multi-qubit quantum models as function approximators for univariate and multivariate functions, respectively. For the single-qubit quantum model experiment, as shown in Fig. 5 (a), we adopted a damping function $f(x) = \sin(5x)/5x$ as target function. The dataset consists of 300 data points uniformly sampled from the interval $x \in (0, 3]$. For the multi-qubit quantum model experiment, as shown in Fig. 5 (b), we adopted a damping function $f(x) = \sin(5x_1)/5x_1 + \sin(5x_2)/5x_2$ as target function. The dataset consists of 300 data points uniformly sampled from the interval $x_1, x_2 \in (0, 3]$. The summarized information can be found in Table 2. For each experiment, we compared NN, QNN, and HQNN. Regarding the NN, we fixed the number of nodes of each layer as 2 to set a similar number of parameters compared to quantum models. With respect to the HQNN, we investigate the effect of the classical embedding network in HQNN. Our findings allow us to contrast the performance of QNNs with NNs, focusing on the number of parameters required and the resulting errors.

Table 3 displays the univariate function approximation performance of the NN, QNN, and HQNN, where # is the number of, C Params denote classical parameters, and Q Params means quantum parameters. For the NN, the performance decreases as the layer increases. In this process, we fixed the number of nodes of each layer as two. These results imply that it is critical to balance the dimensions of the network between width and depth. Note that these results are similar to model scaling [66]. To demonstrate these characteristics, we fixed the number of layers as 5 and only increased the number of nodes. As shown in Fig. 6 (a), the large number of nodes can obtain better performance, where Node = 2, 8, 16, 32 consists of 25, 241, 865, 3265 classical parameters, respectively. In contrast, QNN has a single-qubit model, which means a fixed width. Despite the single-qubit

Table 3 Univariate Function Approximation Performance of the NN, QNN, and HQNN

Model	# Qubit	Layer	# C Params.	# Q Params.	MSE
NN	–	1	2	–	0.0794 ± 0.0018
		5	25	–	0.0896 ± 0.0370
		10	55	–	0.1055 ± 0.0298
QNN	1	1	–	3	0.0818 ± 0.0355
		5	–	15	0.0314 ± 0.0477
HQNN	1	1	4	3	0.0128 ± 0.0145
		5	4	15	0.0002 ± 0.0018
HQNN without Cin	1	1	2	3	0.0213 ± 0.0149
		5	2	15	0.0003 ± 0.0006
NN	–	5	241	–	0.0171 ± 0.0319
		5	865	–	0.0090 ± 0.0202
		5	3265	–	0.0067 ± 0.0148

**Figure 6** Function approximation performance of the classical and quantum models for a simple function

model, the performance of the QNN is improved as the layer increases. Regarding HQNN, although HQNN with single layer has small classical parameters and quantum parameters, it can achieve better performance than NN comprising of 241 parameters and QNN with 5 layers. In addition, compared with the HQNN with and without the classical embedding network with a single layer, we can verify that the synergy of the combination input NN and QNN is significant. In particular, HQNN can achieve dramatically high performance than QNN as the number of layers increases as shown in Fig. 6 (b). These results

Table 4 Multivariate Function Approximation Performance of the NN, QNN, and HQNN

Model	# Qubit	Layer	# C Params.	# Q Params.	MSE
NN	–	1	3	–	0.0669 ± 0.0024
		5	77	–	0.0735 ± 0.0191
		10	57	–	0.0754 ± 0.0065
QNN	2	1	–	6	0.0367 ± 0.0174
		5	–	30	0.0045 ± 0.0056
HQNN	2	1	9	6	0.0010 ± 0.0011
		5	9	30	0.0005 ± 0.0022
HQNN without Cin	2	1	5	6	0.0049 ± 0.0149
		5	5	30	0.0011 ± 0.0014
NN	–	5	3297	–	0.0144 ± 0.0216
		5	12,737	–	0.0124 ± 0.0220
		5	50,049	–	0.0104 ± 0.0197

show that the hybrid quantum architecture might perform better with a smaller number of parameters than the NN and QNN for univariate function approximation.

Table 4 exhibits the multivariate function approximation performance of the NN, QNN, and HQNN. For the NN, the performance decreases as the layer increases. As shown in Fig. 6 (c), we can verify that these results are the same as the univariate function approximation result of the NN, where Node = 2, 32, 64, 128 consists of 77, 3297, 12,737, 50,049 classical parameters, respectively. Regarding the quantum model, hybrid models show higher performance than others, where HQNN outperforms others as shown in Fig. 6 (d). From these results, we can demonstrate the capability of HQNN for function approximation.

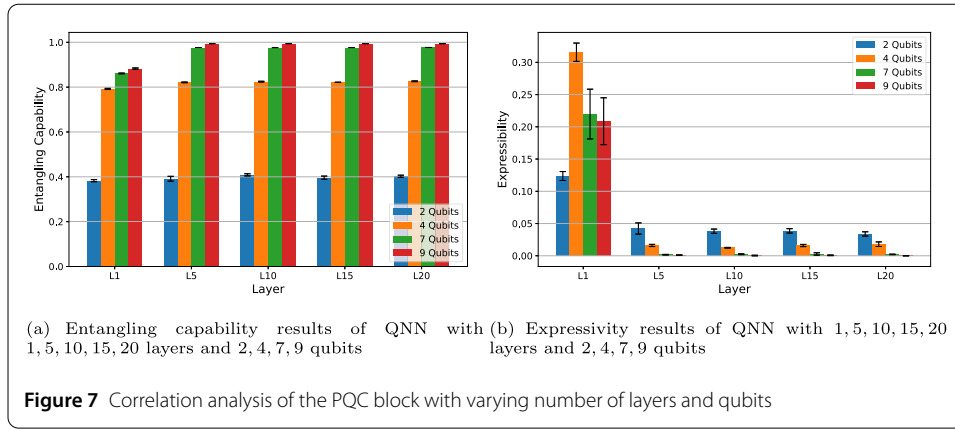
5.3 Analysis of layer and qubit counts effect

To investigate the correlation between the layer and qubit counts and the performance of the PQC block generated by Definition 4, which is a component of the proposed HQNN, we introduce the expressibility and entangling capability [67]. The expressibility and entangling capability metrics are commonly used to evaluate the performance of quantum circuits. Expressibility is known to be strongly correlated with trainability, indicating how effectively a quantum model can explore the parameter space. Meanwhile, high entangling capability reflects the circuit's ability to efficiently represent complex quantum states, which is essential for solving high-dimensional tasks [67]. Therefore, quantum circuits with strong expressibility and high entangling capability are expected to exhibit superior performance. Based on these quantitative metrics, an appropriate number of qubits and circuit layers can be determined prior to applying the full hybrid architecture.

The expressibility of a quantum circuit refers to its capacity to generate quantum states that are uniformly distributed over the Hilbert space. Sim et al. [67] proposed a quantitative measure of this property by comparing the distribution of state fidelities produced by a quantum circuit with the distribution of fidelities obtained from Haar-random states. Formally, expressibility is defined as:

$$\text{Expressibility} = D_{KL} \left(\hat{P}_{QC}(F(\Theta)) || P_{\text{Haar}}(F) \right), \quad (18)$$

where D_{KL} denotes the Kullback-Leibler (KL) divergence [68]. The term $\hat{P}_{QC}(F(\Theta))$ with parameter space Θ represents the estimated probability distribution of fidelities derived



from sampling states generated by the quantum circuit, and $P_{\text{Haar}}(F) = (N - 1)(1 - F)^{N-2}$ is the analytical fidelity distribution corresponding to Haar-random states. Here, $F = |\langle \psi_\theta | \psi_\phi \rangle|^2$ denotes the fidelity between two quantum states, and N is the dimension of the Hilbert space. Consequently, expressibility reaches zero when the two distributions match exactly. Thus, a lower value indicates that the circuit better approximates a Haar-random ensemble, implying greater expressivity and generalization potential.

The entangling capability of a quantum circuit is quantified by the average Meyer–Wallach entanglement [69]:

$$\text{Entangling Capability} = \frac{1}{|S|} \sum_{\theta_i \in S} \mathcal{Q}(|\psi_{\theta_i}\rangle), \quad (19)$$

where $\mathcal{Q}$ denotes the Meyer–Wallach entanglement measure, $S = \{\theta_i\}$ represents the set of sampled circuit parameters. A circuit that produces only separable (product) states yields an entangling capability of 0, while one that consistently generates highly entangled states approaches a value of 1.

As shown in Fig. 7, the entangling capability and expressibility of the PQC block are analyzed by varying the number of layers and qubits. Each configuration is evaluated over 4 independent runs using 1000 shots per run, and the results are reported as the mean with standard deviation. In Fig. 7(a), the entangling capability increases consistently with the number of qubits across all layer settings. For a fixed number of qubits, deeper circuits (with more layers) also exhibit higher entangling capability. Notably, the PQC block with 9 qubits demonstrates significantly stronger entanglement compared to the 2-qubit configuration, approaching the theoretical maximum. Furthermore, configurations with more qubits and layers tend to show lower variance, indicating more stable entanglement generation. Regarding the expressibility results shown in Fig. 7(b), the expressibility of the PQC block increases with both the number of qubits and the number of layers. Notably, the PQC block with 9 qubits approaches maximum expressibility with a low standard deviation as the circuit depth increases.

These results provide a foundation for designing the HQNN with an appropriate number of qubits and layers. Given that the quantum circuit achieves near-maximal expressibility and entangling capability, the remaining performance of the HQNN largely depends on the choice of encoding scheme. Nevertheless, the number of qubits and circuit depth

must be carefully selected to ensure the model's feasibility on NISQ devices, where hardware limitations such as decoherence and qubit counts are significant constraints.

5.4 Analysis of the effect of the encoding scheme

To evaluate the effectiveness of different encoding strategies in our HQDeepDTAF model for protein–ligand binding affinity prediction, we compare prediction accuracy across various configurations involving different embedding types (pure amplitude, NN-Amplitude, and NN-Angle), qubit counts, and circuit depths.

Table 5 summarizes the performance across these settings. The HQDeepDTAF model receives a 384-dimensional input vector, requiring 9 qubits when using pure amplitude embedding. Hence, the pure amplitude model is evaluated at 9 qubits across increasing circuit depths (layers 1 to 20). As shown, both increased qubit counts and deeper circuits generally lead to better performance across all embedding types. However, the pure amplitude embedding consistently underperforms compared to its hybrid counterparts, especially at lower depths. From a circuit complexity perspective, both the pure amplitude and NN-Amplitude embeddings share a similar depth, formulated as $O(\text{poly}(2^n)L + L(3 + n))$, where n is the number of qubits and L is the circuit depth. This overhead arises from the quantum embedding scheme used in both methods. In contrast, NN-Angle embedding exhibits significantly shallower circuit depth, approximated as $O(L + L(3 + n))$, as it does not require exponentially scaling amplitude encoding. Moreover, NN-Angle embedding yields comparable or even superior performance to NN-Amplitude while using fewer classical parameters. For instance, the 9-qubit NN-Angle model with 20 layers achieves an MAE of 1.0821, outperforming NN-Amplitude (1.0856) and significantly outperforming the pure amplitude embedding (1.2984) under identical qubit and layer conditions.

These findings confirm that NN-Angle embedding with 9 qubits is a particularly efficient and scalable configuration for HQDeepDTAF, offering a favorable trade-off among circuit depth, parameter efficiency, and predictive accuracy, making it well-suited for NISQ devices. In addition, from the perspective of the UAT, the key distinction between pure amplitude embedding and hybrid approaches (NN-Amplitude and NN-Angle) is the presence of a classical embedding network. While pure amplitude directly encodes the input into a quantum state, it lacks a trainable mechanism to adapt the input representation. In contrast, NN-based embeddings use a classical network to transform the input into a task-specific latent space before quantum encoding. This added expressivity enhances the model's approximation properties, as reflected in the consistently lower MAEs of hybrid methods. For example, at layer 1, NN-Amplitude achieves 1.1859 vs. 1.4018 for pure amplitude. As layers deepen, this performance gap persists, underscoring the classical embedding's essential role in enabling HQNNs to effectively approximate complex functions.

5.5 Ablation study on HQNN components

To evaluate the effectiveness of the quantum component in the proposed HQDeepDTAF architecture, we conducted an ablation study across five configurations, as presented in Table 6. First, we used a fully classical NN (denoted as Classical NN) as a baseline. Then, we examined two variants of the HQNN architecture based on different quantum embedding methods: 1) NN-Amplitude and 2) NN-Angle. For each embedding type, we considered two settings: one with frozen quantum parameters (Freezing), and the other with trainable quantum parameters (Training).

Table 5 Comparison of the Performance of HQDeepDTAF with varying embedding schemes, layer counts, and qubit counts

Embedding		# Qubit	Layer	Circuit depth	# C param.	# Q param.	MAE
Pure	Amplitude	9	1	$O(\text{poly}(2^9) + 12)$	96,551	27	1.4018 ± 0.0500
			5	$O(\text{poly}(2^9) \cdot 5 + 60)$	96,551	135	1.3835 ± 0.0464
			10	$O(\text{poly}(2^9) \cdot 10 + 120)$	96,551	270	1.3977 ± 0.0246
			15	$O(\text{poly}(2^9) \cdot 15 + 180)$	96,551	405	1.3199 ± 0.0639
			20	$O(\text{poly}(2^9) \cdot 20 + 240)$	96,551	540	1.2984 ± 0.0958
Hybrid	NN-Amplitude	2	1	$O(\text{poly}(2^2) + 4)$	98,084	6	1.1739 ± 0.0309
			5	$O(\text{poly}(2^2) \cdot 5 + 20)$	98,084	30	1.1700 ± 0.0315
			10	$O(\text{poly}(2^2) \cdot 10 + 40)$	98,084	60	1.1642 ± 0.0245
			15	$O(\text{poly}(2^2) \cdot 15 + 60)$	98,084	90	1.1747 ± 0.0057
			20	$O(\text{poly}(2^2) \cdot 20 + 80)$	98,084	120	1.1684 ± 0.0093
		4	1	$O(\text{poly}(2^4) + 6)$	102,706	12	1.1743 ± 0.0408
			5	$O(\text{poly}(2^4) \cdot 5 + 30)$	102,706	60	1.1704 ± 0.0217
			10	$O(\text{poly}(2^4) \cdot 10 + 60)$	102,706	120	1.1647 ± 0.0179
			15	$O(\text{poly}(2^4) \cdot 15 + 90)$	102,706	180	1.1654 ± 0.0150
			20	$O(\text{poly}(2^4) \cdot 20 + 120)$	102,706	240	1.1623 ± 0.0514
		7	1	$O(\text{poly}(2^7) + 10)$	145,829	21	1.1937 ± 0.0241
			5	$O(\text{poly}(2^7) \cdot 5 + 50)$	145,829	105	1.1771 ± 0.0234
			10	$O(\text{poly}(2^7) \cdot 10 + 100)$	145,829	210	1.1684 ± 0.0217
			15	$O(\text{poly}(2^7) \cdot 15 + 150)$	145,829	315	1.1427 ± 0.0092
			20	$O(\text{poly}(2^7) \cdot 20 + 200)$	145,829	510	1.1041 ± 0.0188
		9	1	$O(\text{poly}(2^9) + 12)$	145,831	27	1.1859 ± 0.0396
			5	$O(\text{poly}(2^9) \cdot 5 + 60)$	145,831	135	1.1624 ± 0.0194
			10	$O(\text{poly}(2^9) \cdot 10 + 120)$	145,831	270	1.1432 ± 0.0099
			15	$O(\text{poly}(2^9) \cdot 15 + 180)$	145,831	405	1.1350 ± 0.0109
			20	$O(\text{poly}(2^9) \cdot 20 + 240)$	145,831	540	1.0856 ± 0.0216
Hybrid	NN-Angle	2	1	$O(3)$	97,314	6	1.1687 ± 0.0497
			5	$O(15)$	97,314	30	1.1716 ± 0.2515
			10	$O(30)$	97,314	60	1.1652 ± 0.0067
			15	$O(45)$	97,314	90	1.1658 ± 0.0783
			20	$O(60)$	97,314	120	1.1669 ± 0.0387
		4	1	$O(5)$	98,086	12	1.1724 ± 0.0088
			5	$O(25)$	98,086	60	1.1555 ± 0.0277
			10	$O(50)$	98,086	120	1.1488 ± 0.0278
			15	$O(75)$	98,086	180	1.1337 ± 0.0371
			20	$O(100)$	98,086	240	1.1302 ± 0.0411
		7	1	$O(11)$	99,244	21	1.1890 ± 0.0305
			5	$O(55)$	99,244	105	1.1641 ± 0.0224
			10	$O(110)$	99,244	210	1.1404 ± 0.0184
			15	$O(165)$	99,244	315	1.1150 ± 0.0190
			20	$O(220)$	99,244	420	1.0946 ± 0.0133
		9	1	$O(13)$	100,016	27	1.1794 ± 0.0330
			5	$O(11)$	100,016	135	1.1651 ± 0.0159
			10	$O(55)$	100,016	270	1.1350 ± 0.0220
			15	$O(195)$	100,016	405	1.1138 ± 0.0154
			20	$O(260)$	100,016	540	1.0821 ± 0.0049

Table 6 Comparison of prediction performance and parameter allocation across NN, HQNN with frozen quantum layers, and HQNN with trainable quantum layers

Models	# Qubit	# C param.	# Q param.	Layer	MAE
Classical NN	–	101,336	–	20	1.205 ± 0.202
NN-Amplitude (Freezing)	9	145,831	540	20	1.167 ± 0.033
NN-Amplitude (Training)	9	145,831	540	20	1.086 ± 0.039
NN-Angle (Freezing)	9	100,016	540	20	1.191 ± 0.012
NN-Angle (Training)	9	100,016	540	20	1.082 ± 0.004

All models were configured to have the same number of layers (20) to ensure a fair comparison. The results show that both HQNN variants with trainable quantum layers outperform their frozen counterparts and the classical NN baseline. Notably, NN-Angle (Training) achieved the lowest MAE (1.082 ± 0.004), demonstrating the effectiveness of learning quantum parameters. Even in the freezing settings, both quantum-enhanced models achieved better or comparable performance to the classical NN, indicating that the QNN contributes to representation power even without being trained. This outcome aligns with the findings reported in [70].

In addition, the QNN components in both NN-Amplitude and NN-Angle variants require only 540 quantum parameters, while the classical NN baseline requires 1215 parameters for comparable depth. This highlights the parameter efficiency and scaling advantage of HQNN. As the input dimension increases, this advantage becomes more pronounced, as discussed further in Sect. 6.

5.6 Analysis of binding affinity prediction

To evaluate the effectiveness of the proposed HQDeepDTAF models in predicting protein-ligand binding affinity, we conduct a comprehensive comparison with classical baselines in terms of prediction accuracy and model complexity, as summarized in Table 7. The evaluation metrics include MAE, RMSE, R , SD, and CI, along with the number of classical and quantum parameters.

Among the classical models, DeepDTAF achieves the best overall performance, recording the lowest MAE (1.109) and a competitive RMSE (1.404), while using only 0.15M classical parameters. This significantly outperforms other baselines such as DeepDTA (MAE = 1.208, 1.52M parameters), Pafnucy (MAE = 1.337, 15.8M), TopologyNet (MAE = 1.195, 33.4M), and AEScore (MAE = 1.359). These results highlight DeepDTAF's architectural efficiency, achieving high accuracy with significantly fewer parameters. The proposed HQDeepDTAF models further explore hybrid quantum-classical architectures. Notably, the HQDeepDTAF variants (NN-Amplitude and NN-Angle) that incorporate a classical embedding network and quantum embedding consistently outperform the pure amplitude-based HQDeepDTAF across all accuracy metrics. Specifically, the HQDeepDTAF-NN-Angle model with 9 qubits achieves the best performance, yielding an MAE of 1.082, RMSE of 1.368, and the highest correlation coefficient ($R = 0.783$), along with the lowest prediction variance ($SD = 1.355$).

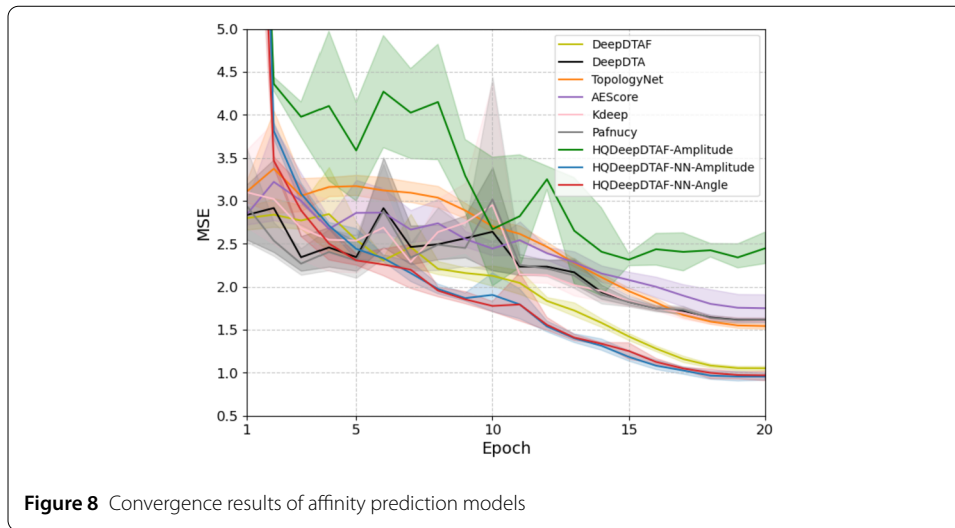
Compared to other hybrid variants and classical baselines, the HQDeepDTAF-NN-Angle model demonstrates not only improved accuracy but also favorable parameter efficiency. It uses fewer classical parameters (100K) than HQDeepDTAF-NN-Amplitude (145K) and classical models like DeepDTA and Pafnucy, suggesting reduced training and inference costs. These results support the effectiveness of combining classical feature processing with quantum circuits, especially through data re-uploading and angle encoding, in achieving accurate and efficient binding affinity prediction under NISQ constraints.

5.7 Analysis of convergence speed

To assess the training efficiency of each model, we further analyze the convergence behavior as illustrated in Fig. 8. Most models converge within 20 epochs, demonstrating stable training dynamics. Notably, HQDeepDTAF-Amplitude and AEScore exhibit larger variance during training, indicating higher sensitivity to initialization or learning conditions. In addition, despite reproducing the original architecture and training

Table 7 Comprehensive Comparison with Qubit, Quantum Parameter, Classical Parameter, Prediction Accuracy of the Proposed HQDeepDTAF and Classical Counterparts on Protein-Ligand Binding Affinity Prediction of the PDBbind Dataset

Models	# Qubit	# C param.	# Q param.	MAE	RMSE	CI	SD	R
DeepDTAF [13]	–	154,114	–	1.109 ± 0.041	1.404 ± 0.049	0.785 ± 0.010	1.390 ± 0.056	0.769 ± 0.022
DeepDTA [34]	–	1,523,396	–	1.208 ± 0.016	1.421 ± 0.019	0.775 ± 0.004	1.356 ± 0.019	0.740 ± 0.008
Pafnucy [8]	–	15,859,081	–	1.337 ± 0.015	1.573 ± 0.013	0.770 ± 0.004	1.559 ± 0.016	0.750 ± 0.007
TopologyNet [10]	–	33,425,345	–	1.195 ± 0.032	1.406 ± 0.039	0.785 ± 0.011	1.392 ± 0.032	0.769 ± 0.024
AE-Score [58]	–	155,131	–	1.359 ± 0.032	1.705 ± 0.036	0.726 ± 0.010	1.685 ± 0.042	0.632 ± 0.024
Kdeep [59]	–	132,954,465	–	1.182 ± 0.017	1.390 ± 0.022	0.786 ± 0.004	1.376 ± 0.021	0.771 ± 0.009
HQDeepDTAF-Amplitude	9	96,551	540	1.298 ± 0.084	1.631 ± 0.110	0.773 ± 0.006	1.449 ± 0.031	0.746 ± 0.013
HQDeepDTAF-NN-Amplitude	7	145,829	420	1.104 ± 0.042	1.388 ± 0.056	0.791 ± 0.010	1.373 ± 0.058	0.776 ± 0.024
	9	145,831	540	1.086 ± 0.039	1.369 ± 0.033	0.794 ± 0.007	1.366 ± 0.032	0.779 ± 0.013
	7	99,244	420	1.095 ± 0.013	1.380 ± 0.016	0.790 ± 0.003	1.373 ± 0.020	0.776 ± 0.007
HQDeepDTAF-NN-Angle	9	100,016	540	1.082 ± 0.004	1.368 ± 0.006	0.792 ± 0.001	1.355 ± 0.008	0.783 ± 0.003



settings, both AEScore and Pafnucy show signs of overfitting in our experiments. Although the HQDeepDTAF-Amplitude model exhibits higher loss variance across runs, it achieves the fastest convergence speed among all models. However, when classical networks are integrated into the quantum model, such as in HQDeepDTAF-NN-Amplitude and HQDeepDTAF-NN-Angle, the convergence speed does not show a clear advantage over classical baselines in terms of the number of epochs. This suggests that incorporating hybrid quantum-classical structures does not necessarily result in faster convergence. This observation is similar with the findings reported in [15]. Nevertheless, this does not imply that quantum models lack training efficiency. It is important to note that all HQDeepDTAF variants use significantly fewer trainable parameters compared to their classical counterparts. As such, they require less computational cost for parameter updates, making them more efficient in terms of training resource consumption. These results suggest that HQDeepDTAF models are not only accurate but also parameter-efficient.

5.8 Analysis of noise effects

To develop the quantum algorithms considering NISQ devices, it is essential to take into account the impact of noise. In these devices, noise affects quantum states during unitary operations and measurements. In our experiments, we evaluate robustness to noise using two widely adopted noise models: depolarizing noise and amplitude damping noise a widely adopted method for noise simulation [15, 71]. We simulate noise at rates of 0.001, 0.01, 0.1, and 0.2. All HQDeepDTAF models use 9 qubits with 20 layers. For the proposed models, we set the number of qubits as 9.

Table 8 summarizes the prediction performance (MAE) under varying noise conditions. In the noiseless setting, the HQDeepDTAF-Hybrid models (NN-Amplitude and NN-Angle) outperform the classical baseline DeepDTAF. As noise is introduced, a gradual degradation in prediction accuracy is observed across all HQDeepDTAF models. This is due to the impact of noise on VQC gradient calculations, which is further explained in Appendix B. Under both noise types, the HQDeepDTAF-NN-Angle model consistently demonstrates the highest robustness, maintaining relatively low MAE values even as the error rate increases. At error rates of 0.001 and 0.01, all hybrid models (NN-Amplitude and NN-Angle) still outperform DeepDTAF. However, when the error rate reaches 0.1 or

Table 8 Prediction performance (MAE) of HQDeepDTAF models and classical counterpart under depolarizing and amplitude damping noise with varying noise rate

Models	Depolarizing noise in model					Amplitude damping noise in model				
	Noiseless	0.001	0.01	0.1	0.2	Noiseless	0.001	0.01	0.1	0.2
DeepDTAF	1.109	–	–	–	–	1.109	–	–	–	–
HQDeepDTAF-Amplitude	1.298	1.325	1.347	1.424	1.489	1.298	1.306	1.335	1.433	1.459
HQDeepDTAF-NN-Amplitude	1.086	1.096	1.108	1.136	1.140	1.086	1.110	1.144	1.140	1.153
HQDeepDTAF-NN-Angle	1.082	1.090	1.096	1.126	1.133	1.082	1.083	1.099	1.118	1.122

higher, all HQDeepDTAF models are affected significantly, and their performance drops below that of DeepDTAF.

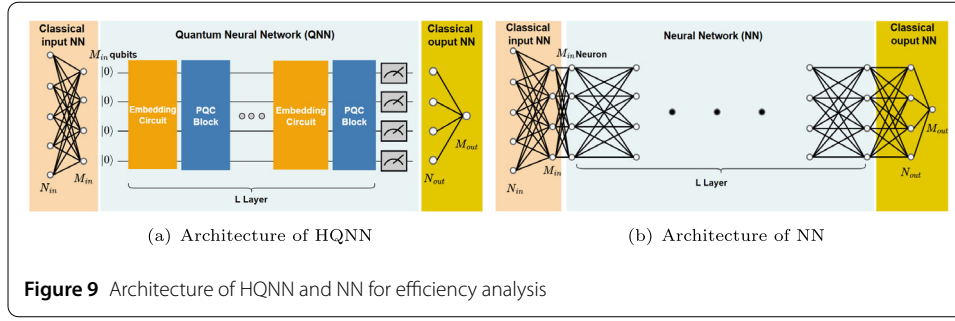
These results demonstrate that HQDeepDTAF-Hybrid models perform better than DeepDTAF when no noise is present. Nonetheless, the superior performance of hybrid models in low-noise settings demonstrates their potential under realistic noise constraints. It is therefore crucial to assess quantum model behavior under expected noise levels to ensure practical viability on near-term hardware.

6 Discussion

In this section, we discuss the efficiency and feasibility of the proposed model, along with practical considerations for implementation on NISQ devices. The efficiency and feasibility analyses are conducted under a theoretical framework [56]. To complement this, we also provide additional discussions addressing key practical challenges and design considerations relevant to NISQ-era quantum computing [72].

The efficiency of the proposed HQNN model was assessed under two conditions: 1) equal number of layers and 2) equal number of trainable parameters, compared to a classical NN. Under the same number of layers condition, we compared the number of trainable parameters, operation time, and model accuracy. Under the same number of parameter conditions, the comparison was conducted in terms of the number of layers, operation time, and accuracy. In particular, to quantify the cost of a quantum algorithm, the total number of gates and the circuit depth are useful [56]. For instance, N gates can be processed in parallel. However, if N gates must be executed in sequence, N time steps are required. For the accuracy comparison, we evaluate the performance of our model against its classical counterpart using ablation studies. Note that our analysis for efficiency focuses solely on HQNN and NN. This is because DeepDTAF and our proposed model share an identical CNN architecture. To evaluate the model's feasibility on NISQ devices, we evaluated its complexity by considering the quantity of qubits, layers, gates, and parameters, where NISQ devices are considered IBM Quantum's quantum computers [73] for simplicity. We focused solely on the hybrid quantum aspects of the model, disregarding the embedding layer and classical convolution modules. Regarding the number of gates, we discuss the complexity of a circuit by the number of $U(4)$ gates since in many architectures they are more difficult to implement than $U(2)$ gates [74].

Efficiency of HQDeepDTAF As shown in Fig. 9, in the efficiency analysis, for the generality, we consider the classical embedding network and classical regression network as classical input NN and output NN, respectively. That is, the HQNN consists of a classical input NN, QNN, and classical output NN sequentially in Fig. 9(a). For the classical NN, the classical NN comprises a classical input NN, NN, and classical output NN sequentially in Fig. 9(b).



Evaluation of the efficiency of HQNN and classical NN For the HQNN, Given N_{in} input neuron of classical input NN, M_{in} output neuron of classical input NN, N_{out} input neuron of classical output NN, M_{out} output neuron of classical output NN, the computational complexity of classical parts of HQNN can be calculated by $N_{in}M_{in} + N_{out}M_{out}$. Assume that all qubits are measured in QNN, then $N_{out} = M_{in}$. Therefore, the computational complexity can be rewritten as $(N_{in} + M_{out})M_{in}$. For the QNN, we assume it is implemented as a data re-uploading QNN utilizing angle embedding. Given M_{in} -dimensional input to QNN, the required number of qubits is $O(M_{in})$. Due to angle embedding, the number of gate operations for encoding is $O(1)$ for each layer. The circuit depth of the embedding circuit is $O(L)$, where L is the number of layers of QNN. With respect to the execution of quantum encoding circuits, the computational complexity is $O(L)$. For PQC blocks with the number of circuit layers L , the strongly entangling circuits have $3M_{in}L$ $U(2)$ quantum gates and $M_{in}L$ $U(4)$ quantum gates. The required number of parameters for PQC blocks can be calculated as $3M_{in}L$. The circuit depth is $O((3 + M_{in})L)$. Therefore, the total circuit depth of the QNN is $O((4 + M_{in})L)$, which is the same as the total execution of quantum circuits of QNN. The total number of quantum parameters of HQNN can be obtained as $O(3M_{in}L)$. The HQNN was implemented using a combination of classical computers and quantum computers. That is, the operation time can be computed as $O((N_{in} + M_{out})M_{in})\text{CPU}_{time} + O(4L)\text{QPU}_{single} + O(M_{in}L)\text{QPU}_{two}$, where CPU_{time} is process time on classical computer, QPU_{single} denotes the $U(2)$ gate operation time on quantum processing unit (QPU), and QPU_{two} means the $U(4)$ gate operation time on QPU. From these analyses, we can verify that HQNN has a polynomial operation time for layers and qubits.

Consider a classical NN with L layers, where the initial layer has N_{in} input neurons and M_{in} output neurons. Each subsequent hidden layer contains M_{in} neurons, while the final layer has M_{out} neurons. The computational complexity of this NN can be expressed as $O((N_{in} + M_{out})M_{in} + M_{in}^2(L + 2))$. The operation time is calculated by multiplying the computational complexity with the CPU time. Table 9 presents the complexity details for this scenario.

In the scenario with the same number of parameters, the classical NN and HQNN are configured with different numbers of layers to enable a fair comparison without loss of generality, as shown in Table 10. Here, L_{nn} denotes the number of layers in the NN, and L_{qnn} represents the number of layers in the QNN.

Comparison First, we calculated the number of parameters and operation time for the first comparison scenario, which is an equal number of layers. In comparing the

Table 9 Evaluation of efficiency of HQNN and NN under the same number of layers

Model	# Qubit	# C params.	# Q params.	# Operation time
Classical NN	–	$(N_{in} + M_{out})M_{in} + M_{in}^2(L + 2)$	–	$O((N_{in} + M_{out})M_{in} + M_{in}^2(L + 2))CPU_{time}$
HQNN	M_{in}	$(N_{in} + M_{out})M_{in}$	$3M_{in}L$	$O((N_{in} + M_{out})M_{in})CPU_{time} + O(4L)QPU_{single} + O(M_{in}L)QPU_{two}$

Table 10 Evaluation of efficiency of HQNN and NN under the same number of parameters

Model	# Qubit	# C params.	# Q params.	# Operation time
Classical NN	–	$(N_{in} + M_{out})M_{in} + M_{in}^2(L_{nn} + 2)$	–	$O((N_{in} + M_{out})M_{in} + M_{in}^2(L_{nn} + 2))CPU_{time}$
HQNN	M_{in}	$(N_{in} + M_{out})M_{in}$	$3M_{in}L_{qnn}$	$O((N_{in} + M_{out})M_{in})CPU_{time} + O(4L_{qnn})QPU_{single} + O(M_{in}L_{qnn})QPU_{two}$

number of parameters, the number of parameters in the classical NN is consistently larger, except when $L = 1$ and $M_{in} = 1$. With respect to the operation time, we can disregard the $O((N_{in} + M_{out})M_{in})$ term when comparing operation times, as this component is shared. That is, in terms of operation time, HQNN and NN are expressed as $O(4L)QPU_{single} + O(M_{in}L)QPU_{two}$ and $O(M_{in}^2(L + 2))CPU_{time}$, respectively, for comparison. In general, $QPU_{two} > QPU_{single}$ [74], we approximate HQNN operation time as $O((4 + M_{in})L)QPU_{two}$ for simple comparison. To achieve faster execution, the QPU time must satisfy the following conditions:

$$M_{in}^2(L + 2)CPU_{time} \geq (M_{in} + 4)LQPU_{two}, \quad (20)$$

$$\frac{M_{in}^2(L + 2)}{(M_{in} + 4)L}CPU_{time} \geq QPU_{two}. \quad (21)$$

The inequality indicates that the HQNN remains advantageous as M_{in} increases, since the classical model's complexity grows faster than that of the HQNN.

For the comparison of the second scenario, to achieve the same total number of parameters, the number of layers of HQNN can be calculated as follows:

$$3M_{in}L_{qnn} = M_{in}^2(L_{nn} + 2), \quad (22)$$

$$L_{qnn} = \frac{M_{in}(L_{nn} + 2)}{3}. \quad (23)$$

To ensure that the above equation, $M_{in}(L_{nn} + 2)$ must be divisible by 3. This condition is satisfied if either $L_{nn} \equiv 1(\text{mod } 3)$ or $M_{in} \equiv 0(\text{mod } 3)$. This constraint allows for deeper QNN architectures, thereby enhancing the model's expressivity [18]. In terms of operation time, and to ensure a fair comparison without loss of generality, the operation time of the HQNN is approximated as $O((4 + M_{in})L_{qnn})QPU_{two}$. To achieve faster execution, the QPU time must satisfy the following conditions:

$$M_{in}^2(L_{nn} + 2)CPU_{time} \geq (M_{in} + 4)L_{qnn}QPU_{two}, \quad (24)$$

$$M_{in}^2(L_{nn} + 2)CPU_{time} \geq (M_{in} + 4)\frac{M_{in}(L_{nn} + 2)}{3}QPU_{two}, \quad (25)$$

$$\frac{3M_{in}}{M_{in} + 4}CPU_{time} \geq QPU_{two}. \quad (26)$$

It is noteworthy that the HQNN can outperform its classical counterpart in terms of operation time, even when the QPU is slower than the CPU. Specifically, the model remains advantageous if the QPU two-qubit gate time satisfies Eq. (26). This inequality implies that the QPU can be up to $3M_{in}/(M_{in} + 4)$ times slower than the CPU, yet still achieve faster overall execution, owing to the structural efficiency of the HQNN architecture.

In summary, in the scenario with the same number of layers, the classical NN consistently requires more parameters. Moreover, the HQNN can achieve faster execution due to the classical NN's complexity grows faster than that of the HQNN. In the scenario with the same number of parameters, the condition $M_{in}(L_{nn} + 2) \equiv 0 \pmod{3}$ enables deeper QNNs and greater expressivity. In large-scale settings, HQNN can achieve faster operation time than classical NN due to the structural efficiency of the HQNN architecture.

Feasibility of HQDeepDTAF To evaluate the feasibility of the HQDeepDTAF for regression task on the NISQ device, we assume that the input dimension of the classical embedding network is N_{in} , and the number of qubits M_{in} .

1) HQDeepDTAF-NN-Amplitude: Given a classical embedding network with N_{in} input neurons and $2^{M_{in}}$ output neurons, the computational complexity is computed as $O(N_{in}2^{M_{in}})$. Given $2^{M_{in}}$ -dimensional input to QNN, the required number of qubits is $\lceil \log(2^{M_{in}}) \rceil$. For simplicity, we assume that the required number of qubits is M_{in} . Considering L layers and data re-uploading scheme, the circuit depth of the amplitude embedding circuit is $O(\text{poly}(M_{in})L)$. Note that the best known lower bound for number of CNOT gates for amplitude embedding is $\frac{1}{4}(4^{M_{in}} - 3M_{in} - 1)$ [26]. Therefore, the total number of U(4) gates is calculated as $\frac{1}{4}(4^{M_{in}} - 3M_{in} - 1)L + M_{in}L$, where the strongly entangling layer [55] is used as PQC ansatz. Finally, the total circuit depth of the QNN with amplitude embedding is $O(\text{poly}(M_{in})L + (3 + M_{in})L)$. The total number of quantum parameters of QNN can be obtained as $O(3M_{in}L)$. The classical regression network is used for regression task. The output dimension of the classical regression network is 1. Therefore, the computational complexity of the classical regression network can be calculated as $O(M_{in})$.

2) HQDeepDTAF-NN-Angle: In the classical embedding network, the number of output neurons corresponds to $O(M_{in})$, which matches the qubit count in the QNN. The computational complexity of this embedding network is calculated as $O(N_{in}M_{in})$. The circuit depth of the angle embedding circuit is $O(L)$. Therefore, the total circuit depth of the QNN with angle embedding is $O((4 + M_{in})L)$. Finally, the total number of quantum parameters of QNN and the computational complexity of the classical regression network is the same as the HQDeepDTAF-NN-Amplitude.

Comparison Table 11 shows the complexity details of our models. We can verify that the HQDeepDTAF-NN-Amplitude requires significantly long coherence time and polynomial U(4) gates. This implies more difficult to implement in practice. In contrast,

Table 11 Evaluation of Complexity of HQDeepDTAF

Model	# Qubit	# Circuit depth	# U(4) gates	# C params.	# Q params
HQDeepDTAF-NN-Amplitude	M_{in}	$O(\text{poly}(M_{in})L + (3 + M_{in})L)$	$\frac{1}{4}(4^{M_{in}} - 3M_{in} - 1)L + M_{in}L$	$N_{in}2^{M_{in}} + M_{in}$	$3M_{in}L$
HQDeepDTAF-NN-Angle	M_{in}	$O(4 + M_{in})L$	$M_{in}L$	$(N_{in} + 1)M_{in}$	$3M_{in}L$

HQDeepDTAF-NN-Angle is more efficient and feasible in terms of circuit depth, $U(4)$ gates, classical parameter, and quantum parameter. In addition, we prove the feasibility of the HQDeepDTAF-NN-Angle on NISQ devices in the following.

Proof For a view of the availability of IBM Quantum's current quantum systems, the proposed models utilized 9 qubits, which is achievable in current NISQ devices [73]. With respect to the required time to execute the quantum circuits, we first determine the specific circuit depth using the experimental values. For the HQDeepDTAF-NN-Amplitude, a depth for embedding circuit is calculated upper bound $O(2^{M_{in}}L)$ referring [26]. Therefore, given 9 qubits and 20 layers, the circuit depth of HQDeepDTAF-NN-Angle and HQDeepDTAF-NN-Amplitude are calculated as 260 and 10,480, respectively. Currently, most IBM quantum computers support a maximum of 300 circuits [73]. Therefore, the only HQDeepDTAF-NN-Angle model can be implemented in the current IBM quantum computer. $\square$

Practical considerations for NISQ implementation While our analysis is closer to fault-tolerant models [56], practical deployment on NISQ hardware requires additional care due to experimental limitations such as quantum noise, decoherence, restricted qubit connectivity, and native gate sets [72]. Our evaluation does not explicitly incorporate qubit connectivity or quantum error mitigation (QEM) techniques, which we acknowledge as limitations of our current framework. Below, we discuss our model from the perspectives of noise robustness and practical feasibility on real NISQ hardware.

1. *Noise Robustness:* Quantum noise is a central challenge in NISQ computing. To assess the robustness of our model, we conducted noise-aware experiments using hardware-realistic noise models. As shown in our results (Sect. 5.8), the performance of HQDeepDTAF noticeably degrades under noisy conditions. This observation highlights the need for incorporating QEM techniques such as zero-noise extrapolation or probabilistic error cancellation [75] in future implementations. While QEM was not applied in this study, our HQDeepDTAF architecture is compatible with such extensions, and integrating them remains an important direction for improving robustness.
2. *Practical Feasibility:* For NISQ feasibility, we demonstrated that the HQDeepDTAF-NN-Angle is more efficient and feasible in terms of circuit depth. While our analysis assumes universal gate operations, actual hardware supports only native gate sets. By the Solovay–Kitaev theorem [76], any unitary operation can be approximated with a logarithmic-length gate sequence from a universal set, introducing additional overhead not captured in our theoretical model. Thus, we recognize that gate synthesis, connectivity constraints, and compilation must be considered in future hardware-specific implementations.

7 Conclusion

This study proposed the HQDeepDTAF algorithm for protein-binding affinity prediction that can be implemented on NISQ devices. Given the limitations of current quantum devices, including qubit availability, noise, and short coherence time, a hybrid quantum model was developed. In addition, this study empirically explored the potential of HQNNs to serve as substitutes for classical NNs by evaluating their capacity for non-linear function approximation. Through ablation studies, we demonstrated that HQNNs effectively

combine the representational strengths of classical and quantum components, highlighting their practical viability under NISQ constraints. Experimental results showed that HQDeepDTAF-NN-Angle and -Amplitude models outperformed benchmarks in protein-binding affinity prediction in terms of the number of parameters and prediction accuracy. To assess the algorithm's effectiveness, noise simulation using the depolarizing channel and amplitude damping channel was performed, confirming the model's resilience to noise. Nevertheless, it is important to note that excessive noise levels can lead to a decline in model effectiveness, making this a crucial factor to take into account. In the discussion, the study evaluated the efficiency and feasibility of the proposed model, comparing HQNN operation time to classical NN and demonstrating faster performance. Feasibility assessment revealed that only the HQDeepDTAF-NN-Angle model is currently implementable on IBM quantum computers, while the HQDeepDTAF-NN-Amplitude model is not feasible due to NISQ device limitations. Furthermore, while additional quantum layers of our model could enhance its performance, the limitations of current NISQ devices prevent this improvement. Although our study explores the potential for quantum advantage, it is important to note that all results are derived from classical simulations. We do not claim a definitive quantum advantage, and further investigation on real quantum devices is necessary to assess this possibility from various perspectives. Thus, to demonstrate our findings, we will evaluate the HQDeepDTAF models in real quantum computers. Moreover, QEM strategies represent a critical direction for future research toward enhancing noise resilience and strengthening the feasibility of quantum models on NISQ devices.

Appendix A: Decomposition of PQC block

Since the single-qubit rotations R is a $U(2)$, the rows and columns of R are orthonormal, from which it follows that there exist real numbers α , β , γ , and δ such that follows [77]:

$$R(\alpha, \beta, \gamma, \delta) = e^{i\delta} \begin{pmatrix} e^{i\beta} \cos(\alpha) & e^{i\gamma} \sin(\alpha) \\ -e^{-i\gamma} \sin(\alpha) & e^{-i\beta} \cos(\alpha) \end{pmatrix}, \quad (\text{A.1})$$

where δ denotes the overall phase factors, which is commonly neglected due to difficulty to physically measure. Therefore, the single-qubit rotations can be considered only three trainable parameters per gate, $R(\alpha, \beta, \gamma)$.

In this paper, we decompose the single-qubit rotations $R \in \mathbf{SU}(2)$ into the product of three rotations [62], which is obtained as follows:

$$\begin{aligned} R(\alpha, \beta, \gamma) &= R_z(\gamma)R_y(\beta)R_z(\alpha) \\ &= \begin{pmatrix} e^{-i(\frac{\alpha+\gamma}{2})} \cos(\frac{\beta}{2}) & -e^{i(\frac{\alpha-\gamma}{2})} \sin(\frac{\beta}{2}) \\ e^{-i(\frac{\alpha-\gamma}{2})} \sin(\frac{\beta}{2}) & e^{i(\frac{\alpha+\gamma}{2})} \cos(\frac{\beta}{2}) \end{pmatrix}. \end{aligned} \quad (\text{A.2})$$

Appendix B: Noise effect for gradient

For an given initial quantum state ρ with a VQC $U(\theta)$, the expectation value is represented as follows:

$$\langle \hat{B}(\theta) \rangle = \text{Tr}(\hat{B}U(\theta)\rho U^\dagger(\theta)),$$

where $\hat{B}$ is the measurement of a final observable $\hat{B}$.

For the optimization process, the gradients of VQCs are computed based on the parameter-shift rule as follows:

$$\nabla_{\theta} \langle \hat{B} \rangle(\theta) = \frac{1}{2} \left[\langle \hat{B} \rangle(\theta + \frac{\pi}{2}) - \langle \hat{B} \rangle(\theta - \frac{\pi}{2}) \right].$$

In the noisy channel $\mathcal{E}$, the expectation value applied noise on the VQCs is rewritten as follows:

$$\begin{aligned} \langle \hat{B}(\theta) \rangle &= \text{Tr}(\hat{B} \mathcal{E} [U(\theta) \rho U^{\dagger}(\theta)]) \\ &= \text{Tr}(\hat{B}' [U(\theta) \rho U^{\dagger}(\theta)]) \\ &= \langle \hat{B}' \rangle(\theta), \end{aligned}$$

where $\mathcal{E}$ denotes the noisy channel. $\hat{B}' = \mathcal{E}^{\dagger}[\hat{B}]$ means a new observable. This impacts the parameter-shift rule, the gradient of the VQCs is rewritten as follows:

$$\nabla_{\theta} \langle \hat{B}' \rangle(\theta) = \frac{1}{2} \left[\langle \hat{B}' \rangle(\theta + \frac{\pi}{2}) - \langle \hat{B}' \rangle(\theta - \frac{\pi}{2}) \right].$$

Abbreviations

AI, Artificial Intelligence; CNNs, Convolutional Neural Networks; CPU, Central Processing Unit; CI, Concordance Index; FC, Fully Connected; FRQI, Flexible Representation of Quantum Image; HQDeepDTAF, Hybrid Quantum DeepDTAF; HQNN, Hybrid Quantum Neural Network; KL, Kullback-Leibler; MAE, Mean Absolute Error; ML, Machine Learning; MSE, Mean Squared Error; NISQ, Noisy Intermediate-Scale Quantum; NNs, Neural Networks; PDBbind, Protein Data Bank Bind; PQC, Parameterized Quantum Circuit; QML, Quantum Machine Learning; QNNs, Quantum Neural Networks; QPU, Quantum Processing Unit; R, Pearson Correlation Coefficient; ReLU, Rectified Linear Unit; RMSE, Root Mean Square Error; SD, Standard Deviation; SDF, Structure Data Format; SMILES, Simplified Molecular Input Line Entry System; SSEs, Secondary Structure Elements; UAP, Universal Approximation Property; UAT, Universal Approximation Theorem.

Acknowledgements

Not applicable.

Author contributions

S.G.J. contributed to the methodology, implementation, and experiment, and prepared the manuscript. K.H.M. contributed to the experiment and prepared the manuscript. W.J.H. supervised the project. All authors reviewed the manuscript.

Funding information

This work was supported by Quantum Computing based on Quantum Advantage challenge research (RS-2024-00408613) through the National Research Foundation of Korea (NRF) funded by the Korean government (Ministry of Science and ICT (MSIT)), and from Busan Regional Innovation System & Education (RISE) Project, supported by the Ministry of Education and Busan Metropolitan City. We would like to thank for support from Creation of the quantum information science R&D ecosystem (based on human resources) through the National Research Foundation of Korea (NRF) funded by the Korean government (MSIT) (RS-2023-00256050), and from the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (RS-2024-00336962). We acknowledge additional support from the Institute of Information and communications Technology Planning and Evaluation (IITP) under the Artificial Intelligence Convergence Innovation Human Resources Development (IITP-2025-RS-2023-00254177) grant funded by the Korea government (MSIT).

Data availability

Our code is available at <https://github.com/drmoon-1st/HQDeepDTAF>.

Declarations

Competing interests

The authors declare no competing interests.

References

1. Kitchen DB, Decornez H, Furr JR, Bajorath J. Docking and scoring in virtual screening for drug discovery: methods and applications. *Nat Rev Drug Discov*. 2004;3(11):935–49.
2. Zheng L, Fan J, Mu Y. Onionnet: a multiple-layer intermolecular-contact-based convolutional neural network for protein–ligand binding affinity prediction. *ACS Omega*. 2019;4(14):15956–65.
3. Burley SK, Berman HM, Bhikadiya C, Bi C, Chen L, Di Costanzo L, Christie C, Dalenberg K, Duarte JM, Dutta S, et al. Rcsb protein data bank: biological macromolecular structures enabling research and education in fundamental biology, biomedicine, biotechnology and energy. *Nucleic Acids Res*. 2019;47(D1):464–74.
4. Zhao Y, Sanner MF. Flippedock: docking flexible ligands into flexible receptors. *Proteins, Struct Funct Bioinform*. 2007;68(3):726–37.
5. Guterres H, Im W. Improving protein–ligand docking results with high-throughput molecular dynamics simulations. *J Chem Inf Model*. 2020;60(4):2189–98.
6. Brylinski M. Nonlinear scoring functions for similarity-based ligand docking and binding affinity prediction. Washington: ACS; 2013.
7. Cobanoglu MC, Liu C, Hu F, Oltvai ZN, Bahar I. Predicting drug–target interactions using probabilistic matrix factorization. *J Chem Inf Model*. 2013;53(12):3399–409.
8. Stepniewska-Dziubinska MM, Zielenkiewicz P, Siedlecki P. Development and evaluation of a deep learning model for protein–ligand binding affinity prediction. *Bioinformatics*. 2018;34(21):3666–74.
9. Rezaei M, Li Y, Li X, Li C. Improving the accuracy of protein–ligand binding affinity prediction by deep learning models: benchmark and model. *ChemRxiv*. 2019. <https://doi.org/10.26434/chemrxiv.9866912>.
10. Cang Z, Wei G-W. Topologynet: topology based deep convolutional and multi-task neural networks for biomolecular property predictions. *PLoS Comput Biol*. 2017;13(7):1005690.
11. Zeng Y, Chen X, Luo Y, Li X, Peng D. Deep drug–target binding affinity prediction with multiple attention blocks. *Brief Bioinform*. 2021;22(5):117.
12. Öztürk H, Ozkirimli E, Özgür A. Widedta: prediction of drug–target binding affinity. 2019. arXiv preprint. [arXiv:1902.04166](https://arxiv.org/abs/1902.04166).
13. Wang K, Zhou R, Li Y, Li M. Deepdtaf: a deep learning method to predict protein–ligand binding affinity. *Brief Bioinform*. 2021;22(5):072.
14. Askr H, Elgeldawi E, Aboul Ella H, Elshaier YA, Gomaa MM, Hassanien AE. Deep learning in drug discovery: an integrative review and future challenges. *Artif Intell Rev*. 2023;56(7):5975–6037.
15. Moon K-H, Jeong S-G, Hwang W-J. Qsegrnn: quantum segment recurrent neural network for time series forecasting. *EPJ Quantum Technol*. 2025;12(1):32.
16. Jeong S-G, Do Q-V, Hwang H-J, Hasegawa M, Sekiya H, Hwang W-J. Hybrid quantum convolutional neural networks for UWB signal classification. *IEEE Access*. 2023;11:113726–39.
17. Jeong S-G, Do QV, Hwang W-J. Short-term photovoltaic power forecasting based on hybrid quantum gated recurrent unit. *ICT Express*. 2024;10(3):608–13.
18. Pérez-Salinas A, Cervera-Lierta A, Gil-Fuster E, Latorre JI. Data re-uploading for a universal quantum classifier. *Quantum*. 2020;4:226.
19. Schuld M, Sweke R, Meyer JJ. Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Phys Rev A*. 2021;103:032430.
20. Wu Y, Yao J, Zhang P, Zhai H. Expressivity of quantum neural networks. *Phys Rev Res*. 2021;3(3):032049.
21. Cao Y, Romero J, Aspuru-Guzik A. Potential of quantum computing for drug discovery. *IBM J Res Dev*. 2018;62(6):6–1620.
22. Blunt NS, Camps J, Crawford O, Izsák R, Leontica S, Mirani A, Moylett AE, Scivier SA, Sunderhauf C, Schopf P, et al. Perspective on the current state-of-the-art of quantum computing for drug discovery applications. *J Chem Theory Comput*. 2022;18(12):7001–23.
23. Yu Z, Yao H, Li M, Wang X. Power and limitations of single-qubit native quantum neural networks. *Adv Neural Inf Process Syst*. 2022;35:27810–23.
24. Chen S, Cotler J, Huang H-Y, Li J. The complexity of nisq. *Nat Commun*. 2023;14(1):6001.
25. Jeong S-G, Duc PDA, Do QV, Noh D-I, Tung NX, Chien TV, Pham Q-V, Hasegawa M, Sekiya H, Hwang W-J. Quantum annealing-based sum rate maximization for multi-uav-aided wireless networks. *IEEE Internet Things J*. 2025;12:21225–39.
26. Sun X, Tian G, Yang S, Yuan P, Zhang S. Asymptotically optimal circuit depth for quantum state preparation and general unitary synthesis. *IEEE Trans Comput-Aided Des Integr Circuits Syst*. 2023;42(10):3301–14.
27. LaRose R, Coyle B. Robust data encodings for quantum classifiers. *Phys Rev A*. 2020;102(3):032420.
28. Liu Z, Su M, Han L, Liu J, Yang Q, Li Y, Wang R. Forging the basis for developing protein–ligand interaction scoring functions. *Acc Chem Res*. 2017;50(2):302–9.
29. Cao Y, Guerreschi GG, Aspuru-Guzik A. Quantum Neuron: an elementary building block for machine learning on quantum computers. 2017. arXiv preprint. [arXiv:1711.11240](https://arxiv.org/abs/1711.11240).
30. Torrontegui E, García-Ripoll JJ. Unitary quantum perceptron as efficient universal approximator (a). *Europhys Lett*. 2019;125(3):30004.
31. Maronese M, Destri C, Prati E. Quantum activation functions for quantum neural networks. *Quantum Inf Process*. 2022;21(4):128.
32. Inajetovic MA, Orazi F, Macaluso A, Lodi S, Sartori C. Enabling non-linear quantum operations through variational quantum splines. In: International conference on computational science. Berlin: Springer; 2023. p. 177–92.
33. Lubasch M, Joo J, Moinier P, Kiffner M, Jaksch D. Variational quantum algorithms for nonlinear problems. *Phys Rev A*. 2020;101(1):010301.
34. Öztürk H, Özgür A, Ozkirimli E. Deepdta: deep drug–target binding affinity prediction. *Bioinformatics*. 2018;34(17):821–9.
35. Li Y, Rezaei MA, Li C, Li X. Deepatom: a framework for protein–ligand binding affinity prediction. In: 2019 IEEE international conference on bioinformatics and biomedicine (BIBM). IEEE; 2019. p. 303–10.
36. Karimi M, Wu D, Wang Z, Shen Y. Deepaffinity: interpretable deep learning of compound–protein affinity through unified recurrent and convolutional neural networks. *Bioinformatics*. 2019;35(18):3329–38.

37. Domingo L, Djukic M, Johnson C, Borondo F. Binding affinity predictions with hybrid quantum-classical convolutional neural networks. *Sci Rep*. 2023;13(1):17951.
38. Dong L, Li Y, Liu D, Ji Y, Hu B, Shi S, Zhang F, Hu J, Qian K, Jin X, et al. Prediction of protein-ligand binding affinity by a hybrid quantum-classical deep learning algorithm. *Adv Quantum Technol*. 2023;6(9):2300107.
39. Domingo L, Chehimi M, Banerjee S, Yuxun SH, Konakanchi S, Ogunfowora L, Roy S, Selvarajan S, Djukic M, Johnson C. A hybrid quantum-classical fusion neural network to improve protein-ligand binding affinity predictions for drug discovery. In: 2024 IEEE international conference on quantum computing and engineering (QCE). vol. 02. 2024. p. 126–31.
40. Tüysüz C, Rieger C, Novotny K, Demirköz B, Dobos D, Potamianos K, Vallecorsa S, Vlimant J-R, Forster R. Hybrid quantum classical graph neural networks for particle track reconstruction. *Quantum Mach Intell*. 2021;3:1–20.
41. Senokosov A, Sedykh A, Sagingalieva A, Kyriacou B, Melnikov A. Quantum machine learning for image classification. *Mach Learn: Sci Technol*. 2024;5(1):015040.
42. Ovalle-Magallanes E, Alvarado-Carrillo DE, Avina-Cervantes JG, Cruz-Aceves I, Ruiz-Pinales J. Quantum angle encoding with learnable rotation applied to quantum-classical convolutional neural networks. *Appl Soft Comput*. 2023;141:110307.
43. Sinha A, Macaluso A, Klusch M. Nav-q: quantum deep reinforcement learning for collision-free navigation of self-driving cars. *Quantum Mach Intell*. 2025;7(1):1–20.
44. Aloy P, Russell RB. Structural systems biology: modelling protein interactions. *Nat Rev, Mol Cell Biol*. 2006;7(3):188–97.
45. Lloyd S, Schuld M, Ijaz A, Izaac J, Killoran N. Quantum embeddings for machine learning. 2020. arXiv preprint. [arXiv:2001.03622](https://arxiv.org/abs/2001.03622).
46. Lu Z, Pu H, Wang F, Hu Z, Wang L. The expressive power of neural networks: a view from the width. *Adv Neural Inf Process Syst*. 2017;30.
47. Weininger D. Smiles, a chemical language and information system. 1. Introduction to methodology and encoding rules. *J Chem Inf Comput Sci*. 1988;28(1):31–6.
48. O'Boyle NM, Banck M, James CA, Morley C, Vandermeersch T, Hutchison GR. Open babel: an open chemical toolbox. *J Cheminform*. 2011;3:1–14.
49. Harding-Larsen D, Funk J, Madsen NG, Gharabli H, Acevedo-Rocha CG, Mazurenko S, Welner DH. Protein representations: encoding biological information for machine learning in biocatalysis. *Biotechnol Adv*. 2024;77:108459.
50. Zhang F, Shi W, Zhang J, Zeng M, Li M, Kurgan L. Probselect: accurate prediction of protein-binding residues from proteins sequences via dynamic predictor selection. *Bioinformatics*. 2020;36(Supplement_2):735–44.
51. Magnan CN, Baldi P. Sspro/acpro 5: almost perfect prediction of protein secondary structure and relative solvent accessibility using profiles, machine learning and structural similarity. *Bioinformatics*. 2014;30(18):2592–7.
52. Wang L, Berne B, Friesner R. Ligand binding to protein-binding pockets with wet and dry regions. *Proc Natl Acad Sci USA*. 2011;108(4):1326–30.
53. Chen Y, Guo Q, Liang X, Wang J, Qian Y. Environmental sound classification with dilated convolutions. *Appl Acoust*. 2019;148:123–32.
54. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. p. 770–8.
55. Schuld M, Bocharov A, Svore KM, Wiebe N. Circuit-centric quantum classifiers. *Phys Rev A*. 2020;101(3):032308.
56. Nielsen MA, Chuang IL. Quantum computation and quantum information. Cambridge: Cambridge University Press; 2010.
57. Mitarai K, Negoro M, Kitagawa M, Fujii K. Quantum circuit learning. *Phys Rev A*. 2018;98(3):032309.
58. Meli R, Anighoro A, Bodkin MJ, Morris GM, Biggin PC. Learning protein-ligand binding affinity with atomic environment vectors. *J Cheminform*. 2021;13(1):59.
59. Jiménez J, Skalic M, Martinez-Rosell G, De Fabritiis G. K deep: protein–ligand absolute binding affinity prediction via 3d-convolutional neural networks. *J Chem Inf Model*. 2018;58(2):287–96.
60. Zhao M, Lee W-P, Garrison EP, Marth GT. SSW library: an SIMD Smith-Waterman C/C++ library for use in genomic applications. *PLoS ONE*. 2013;8(12):82138.
61. Loshchilov I, Hutter F. Decoupled Weight Decay Regularization. 2019. arXiv preprint. [arXiv:1711.05101](https://arxiv.org/abs/1711.05101).
62. Bergholm V, Izaac J, Schuld M, Gogolin C, Ahmed S, Ajith V, Alam MS, Alonso-Linaje G, AkashNarayanan B, Asadi A, Arazola JM, Azad U, Banning S, Blank C, Bromley TR, Cordier BA, Ceroni J, Delgado A, Matteo OD, Dusko A, Garg T, Guala D, Hayes A, Hill R, Ijaz A, Isacsson T, Ittah D, Jahangiri S, Jain P, Jiang E, Khandelwal A, Kottmann K, Lang RA, Lee C, Loke T, Lowe A, McKiernan K, Meyer JJ, Montañez-Barrera JA, Moyard R, Niu Z, O'Riordan LJ, Oud S, Panigrahi A, Park C-Y, Polatajko D, Quesada N, Roberts C, Sá N, Schoch I, Shi B, Shu S, Sim S, Singh A, Strandberg I, Soni J, Száva A, Thabet S, Vargas-Hernández RA, Vincent T, Vitucci N, Weber M, Wierichs D, Wiersema R, Willmann M, Wong V, Zhang S, Killoran N. PennyLane: automatic differentiation of hybrid quantum-classical computations. 2022. arXiv preprint. [arXiv:1811.04968](https://arxiv.org/abs/1811.04968).
63. Cohen I, Huang Y, Chen J, Benesty J, Benesty J, Chen J, Huang Y, Cohen I. Pearson correlation coefficient. Noise reduction in speech processing. 2009. 1–4.
64. Cheshier D. Evaluating assay precision. *Clin Biochem Rev*. 2008;29(Suppl 1):23.
65. Pahikkala T, Airola A, Pietilä S, Shakyawar S, Sz wajda A, Tang J, Aittokallio T. Toward more realistic drug–target interaction predictions. *Brief Bioinform*. 2015;16(2):325–37.
66. Tan M, Le Q. Efficientnet: rethinking model scaling for convolutional neural networks. In: International conference on machine learning. PMLR; 2019. p. 6105–14.
67. Sim S, Johnson PD, Aspuru-Guzik A. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *Adv Quantum Technol*. 2019.
68. Kullback S, Leibler RA. On information and sufficiency. *Ann Math Stat*. 1951;22(1):79–86.
69. Meyer DA, Wallach NR. Global entanglement in multiparticle systems. *J Math Phys*. 2002;43(9):4273–8.
70. Henderson M, Shakya S, Pradhan S, Cook T. Quconvolutional neural networks: powering image recognition with quantum circuits. *Quantum Mach Intell*. 2020;2(1):2.
71. Zhao R-X, Shi J, Li X. Qksan: a quantum kernel self-attention network. *IEEE Trans Pattern Anal Mach Intell*. 2024;46(12):10184–95.

72. Bharti K, Cervera-Lierta A, Kyaw TH, Haug T, Alperin-Lea S, Anand A, Degroote M, Heimonen H, Kottmann JS, Menke T, et al. Noisy intermediate-scale quantum algorithms. *Rev Mod Phys*. 2022;94(1):015004.
73. AbuGhanem M. IBM quantum computers: evolution, performance, and future directions. 2024. arXiv preprint. [arXiv: 2410.00916](https://arxiv.org/abs/2410.00916).
74. Malvetti E, Iten R, Colbeck R. Quantum circuits for sparse isometries. *Quantum*. 2021;5:412.
75. Temme K, Bravyi S, Gambetta JM. Error mitigation for short-depth quantum circuits. *Phys Rev Lett*. 2017;119(18):180509.
76. Dawson CM, Nielsen MA. The Solovay-Kitaev algorithm. 2005. arXiv preprint. [arXiv:quant-ph/0505030](https://arxiv.org/abs/quant-ph/0505030).
77. Barenco A, Bennett CH, Cleve R, DiVincenzo DP, Margolus N, Shor P, Sleator T, Smolin JA, Weinfurter H. Elementary gates for quantum computation. *Phys Rev A*. 1995;52(5):3457.

Publisher's note

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

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

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

Submit your next manuscript at ► [springeropen.com](https://www.springeropen.com)