mathematics

IMPACT FACTOR 2.3

CITESCORE 4.0

Article

Hybrid Quantum Neural Network Approaches to Protein–Ligand Binding Affinity Prediction

Maria Avramouli, Ilias K. Savvas, Anna Vasilaki, Andreas Tsipourlianos and Georgia Garani

Special Issue

Quantum Control and Machine Learning in Quantum Technology

Edited by

Dr. Bijita Sarma and Dr. Sangkha Borah

MDPI

https://doi.org/10.3390/math12152372

mathematics

MDPI

Article

Hybrid Quantum Neural Network Approaches to Protein–Ligand Binding Affinity Prediction

Maria Avramouli [1,*], Ilias K. Savvas [1], Anna Vasilaki [2], Andreas Tsipourlianos [1] and Georgia Garani [1]

[1] Department of Digital Systems, University of Thessaly, GAIOPOLIS, 41500 Larissa, Greece; isavvas@uth.gr (I.K.S.); antsipou@uth.gr (A.T.); garani@uth.gr (G.G.)
[2] Laboratory of Pharmacology, Faculty of Medicine, University of Thessaly, BIOPOLIS, 41500 Larissa, Greece; a.vasilaki@uth.gr
* Correspondence: mavramouli@uth.gr

Abstract: Drug repositioning is a less expensive and time-consuming method than the traditional method of drug discovery. It is a strategy for identifying new uses for approved or investigational drugs that are outside the scope of the original medical indication. A key strategy in repositioning approved or investigational drugs is determining the binding affinity of these drugs to target proteins. The large increase in available experimental data has helped deep learning methods to demonstrate superior performance compared to conventional prediction and other traditional computational methods in precise binding affinity prediction. However, these methods are complex and time-consuming, presenting a significant barrier to their development and practical application. In this context, quantum computing (QC) and quantum machine learning (QML) theoretically offer promising solutions to effectively address these challenges. In this work, we introduce a hybrid quantum–classical framework to predict binding affinity. Our approach involves, initially, the implementation of an efficient classical model using convolutional neural networks (CNNs) for feature extraction and three fully connected layers for prediction. Subsequently, retaining the classical module for feature extraction, we implement various quantum and classical modules for binding affinity prediction, which accept the concatenated features as input. Quantum predicted modules are implemented with Variational Quantum Regressions (VQRs), while classical predicted modules are implemented with various fully connected layers. Our findings clearly show that hybrid quantum–classical models accelerate the training process in terms of epochs and achieve faster stabilization. Also, these models demonstrate quantum superiority in terms of complexity, accuracy, and generalization, thereby indicating a promising direction for QML.

Keywords: quantum machine learning; quantum computing; protein–ligand binding affinity; quantum neural network; quantum regression

MSC: 81-04; 81-05

Citation: Avramouli, M.; Savvas, I.K.; Vasilaki, A.; Tsipourlianos, A.; Garani, G. Hybrid Quantum Neural Network Approaches to Protein–Ligand Binding Affinity Prediction. *Mathematics* **2024**, *12*, 2372. https://doi.org/10.3390/math12152372

Academic Editors: Bijita Sarma and Sangkha Borah

Received: 28 June 2024
Revised: 26 July 2024
Accepted: 28 July 2024
Published: 30 July 2024

1. Introduction

The drug discovery process is time-consuming and quite expensive. The development of a new drug can take approximately 12 years, and it is estimated that its average cost, until it reaches the market, is around EUR two billion. The time and cost associated are largely due to the large number of compounds that fail at one or more stages of their development, as it is estimated that only 1 in 5.000 compounds eventually reach the market [1]. Drug repositioning is a less expensive and time-consuming method than the traditional method of drug development. Drug repositioning (also called drug repurposing, reprofiling, or re-tasking) is a strategy for identifying new uses for approved or investigational drugs that are outside the scope of the original medical indication [2]. Given that a significant part of research focuses on pharmaceuticals designed for prevalent diseases, such as cardiovascular

diseases, cancer, and Alzheimer's, the practice of drug repositioning assumes considerable significance in the treatment of rare diseases.

A key strategy in repositioning approved or investigational drugs is determining the binding affinity of these drugs to target proteins [3]. Drugs typically function as ligands, interacting with target proteins. Binding affinity refers to the strength of the interaction between a compound (such as a drug/ligand) and its target (such as a protein/receptor). It quantifies how tightly the molecule binds to the target and is typically expressed by the inhibition constant (Ki) or dissociation constant (Kd). Compounds with higher binding affinities are more likely to produce therapeutic effects. Experimental methods for determining protein–ligand binding affinity are generally considered the most reliable. However, the experimental determination of protein–ligand affinity is often time-consuming and costly. Therefore, there is a growing interest in developing alternative computational methods to accurately estimate protein–ligand binding affinity. More specifically, estimating binding affinity through computational methods can aid in prioritizing suitable targets candidates from a vast pool for subsequent experimental testing, thereby accelerating the drug discovery process.

Taking advantage of advancements in computer-aided drug design, several physics-based computational techniques have emerged for the precise estimation of protein–ligand binding affinity. These methods include molecular dynamics simulations [4] and free energy simulations [5]. Nevertheless, screening large-scale protein–ligand complexes using these methods remains a considerable challenge due to the substantial conventional overheads involved. In contrast to physics-based approaches, numerous traditional machine-learning-based computational methods have been developed to enhance the predictive accuracy of protein–ligand binding affinity. Examples include RFscore, which employs random forest [6], and Pred-binding, which uses Support Vector Machine [7]. However, the effectiveness of such machine-learning-based techniques heavily depends on the creation of efficient manually extracted features, requiring fundamental domain expertise [8]. Deep learning (DL) methods, capable of autonomously learning feature representations from raw inputs without domain expertise [9], have garnered significant interest. However, these methods are complex and time-consuming, presenting a significant barrier to their development and practical application. Furthermore, conventional deep learning models face several challenges when handling data for drug discovery methods: firstly, the data dimensions become significantly large, and, secondly, there is a growing volume of data that must be processed, thereby substantially increasing the computational power requirements.

Quantum computing (QC) and quantum machine learning (QML) theoretically offer promising solutions to effectively address these challenges. Superposition and entanglement, fundamental principles of quantum computers, are used to solve efficiently complex mathematical problems. QC has the potential to speed up machine learning algorithms by executing specific computations, such as matrix inversions and eigen decompositions, at a significantly faster rate compared to classical counterparts [10]. The quantum superposition principle enables more efficient processing of high-dimensional data using fewer quantum computing resources, due to the exponential scaling of the Hilbert space, reducing the critical computational power lack currently faced by DL. Nevertheless, research on QML for predicting binding affinity is significantly lacking, requiring further development. Therefore, this study proposes classical and hybrid quantum–classical neural networks to predict binding affinity, along with corresponding evaluation procedures. The objective is twofold: first, how to combine classical and quantum methods for regression problems; second, how to combine them for binding affinity prediction.

The primary contributions of this study are as follows:

- A classical 1D convolutional neural network (CNN) is proposed to extract protein and ligands features effectively. This does not predicate prior knowledge of complex structures and learn the characteristics of entities in parallel.
- A classical model is proposed to predict binding affinity.

- Three quantum regression modules (Variational Quantum Regression—VQR) combined effectively with the classical 1D CNN to predict binding affinity.
- An integrated framework for implementing and evaluating classical and QML models.

The rest of this paper is structured as follows: Section 2 presents related work and provides some preliminary concepts about quantum computing. Section 3 describes the development of classical and hybrid quantum–classical models. Section 4 introduces the experiment, conducted using classical and hybrid models, evaluation, and results. Finally, Section 5 concludes the paper and outlines future research directions.

2. Related Work—Preliminaries

In this section, the research advancements related to deep learning in binding affinity prediction and the achievements of quantum neural network algorithms in drug discovery and other fields are presented.

2.1. Deep Learning in Prediction of Binding Affinity

Some advancements have been made in the application of DL methods to binding affinity prediction tasks. These DL approaches are categorized based on two types of input data forms: complex-based inputs and non-complex-based inputs [11]. Complex-based inputs include data directly encoded from protein–ligand complex structures, such as feature-embedded 3D grid representations and graph representations of complex. These models are based on experimentally obtained protein–ligand complexes, which are generated in the laboratory and are inherently limited. Consequently, this fact leads to a restricted scope for these models. Non-complex-based inputs refer to data derived from separated protein and ligand information. These methods do not necessitate prior knowledge of complex structures and learn the characteristics of entities in parallel.

In approaches based on complex-based inputs, Cang and Wei [12] introduced TopologyNet, using the element-specific persistent homology (ESPH) method (representing 3D complex geometry with 1D topological invariants) alongside deep CNNs to establish a multichannel topological neural network for predicting protein–ligand binding affinities and protein stability changes upon mutation. Stepniewska-Dziubinska et al. [13] designed the Pafnucy, which encodes the input structure as a 3D grid with 19 features in each voxel, employs 3D convolution to generate a feature map of the representation, and subsequently uses dense layers to predict the binding affinity score. Wang et al. [14] developed DeepDTAF, which employs sequences and structural properties (such as secondary structure elements—SSEs [15]) for protein inputs and Simplified Molecular Input Line Entry System (SMILES) [16] for ligand inputs. Additionally, for protein-binding pockets, it uses discontinuous sequence and secondary structure elements (SSEs). These input features are fed into embedding layers and dilated or traditional convolution layers, followed by three fully connected layers for prediction. Wang et al. [17] introduced a 2D CNN model (OnionNet-2) that employs protein–ligand complexes for training and employs rotation-free residue-atom-specific contacts in multiple distance shells to characterize the protein–ligand interactions. The cloud-based neural network structures PointNet and PointTransformer were used in PointTransformer model [18] for predicting protein–ligand affinity.

On the contrary, non-complex-based input methods do not necessitate prior knowledge of complex structures and have consequently been used in many studies. The first model employing this approach was introduced by Öztürk et al. [19]. This deep learning model, named DeepDTA, uses two CNNs to learn representations from protein sequences and ligand SMILES. Rezaei et al. [20] developed a CNN model named DeepAtom. The central component of DeepAtom lies in the 3D shuffle group, consisting of stacked 3D convolutional layers featuring small kernel sizes and strides. This configuration reduces the parameter count while simultaneously enhancing model depth and complexity. Abbasi et al. [21] developed the DeepCDA model, which combines CNN and LSTM (Long Short-Term Memory) layers into a unified framework for encoding local and global temporal patterns. The model takes as input the protein sequences and SMILES representations

of compounds. A two-sided attention mechanism is employed to fuse the compound and protein descriptors, followed by a Multilayer Perceptron (MLP) for predicting the affinity. Nguyen et al. [22] introduced a model known as GraphDTA. In this approach, drugs are encoded as graphs with a feature map and an adjacency matrix. These representations are processed by Graph Neural Networks (GNNs), including Graph Convolutional Networks (GCNs), Graph Attention Networks (GATs), and Graph Isomorphic Networks (GINs). Meanwhile, proteins are represented using convolutional blocks, and two fully connected layers are employed to predict drug–target affinity. FusionDTA [23] encodes drug molecules as SMILES strings and proteins as word embeddings. The encoder layer employs LSTM layers as fundamental building blocks. Intermediate carriers of drug molecules and proteins are fed into the fusion layer, producing an output carrier representation indicative of binding affinity. MFR-DTA [24] uses amino acid embedding and word embedding to represent protein features. For drug features, it employs functional-connectivity fingerprints (FCFPs) and GNN. The proposed BioMLP and BioCNN modules assist the model in extracting individual features of sequence and SMILES elements, respectively. A fully connected neural network is used to predict drug–target affinity. Jin et al. [25] proposed CAPLA, a model comprising two 1D dilated CNNs for the protein and the ligand, respectively, along with a 1D CNN for the binding pocket. The input representation for a protein and pocket includes the amino acid sequences, the protein's SSEs, and the physicochemical properties of residues. Meanwhile, SMILES represents the input for the ligand. A cross-attention mechanism to jointly extract features from protein-binding pockets and ligands is used. Two fully connected layers and an output layer are employed for predicting binding affinity. Zhu et al. [26] introduced DataDTA, a model that utilizes descriptors of predicted pockets from the third part of application, sequences of proteins, alongside low-dimensional molecular features, and SMILES of compounds, as input variables. The molecular representation of compounds based on algebraic graph features is gathered to complement the input information of targets. A dual-interaction aggregation neural network strategy is implemented to facilitate the effective learning of multiscale interaction features. Additionally, an MLP is employed for predicting affinity.

2.2. Quantum Neural Networks

2.2.1. General Field

Quantum neural networks (QNNs) leverage qubits and gates for computation and learning within quantum-computing environments. The earliest study on QML can be traced back to 1995, when Kak [27] introduced the concept of quantum neural computation. Following this, a plethora of researchers have proposed models based on QNNs. Since then, many QNN studies with theoretical features have been carried out [28–32], resulting in a lack of implementation at the current stage of quantum devices.

In the Noisy Intermediate-Scale Quantum (NISQ) era [33], variational quantum algorithms (VQAs) [34] have emerged as an important research direction, with QNN based on variational quantum circuits garnering widespread notice. One of the earliest implementations of a QNN is presented in [35], namely a quantum perceptron. This quantum algorithm introduces the binary perceptron, demonstrating an exponential advantage in storage resources. Zhao et al. [36] proposed a QNN model based on a swap test, where inputs, outputs, and weights between neurons are all quantum states. The corresponding circuit was designed, and experiments were conducted. Cong et al. [37] introduced a quantum convolutional neural network (QCNN) capable of processing classical image data while yielding results close to classical convolutions, thereby significantly broadening the scope of QNN applications. Pesah et al. [38] demonstrated that by constructing appropriate quantum circuits, QNNs would not be afflicted by the barren plateaus that afflict classical neural networks, thereby showcasing the potential advantages of QNNs. Zhao et al. [39] introduced a hybrid quantum–classical model and multilayer neural network, employing the quantum component to compute neural network parameters, while the output can

be fed into a classical computer. This model achieves a high degree of accuracy in image classification tasks.

Recent mathematical analyses indicate that certain properties of an n-qubit system can be efficiently learned by a quantum machine, while the necessary number of conventional experiments to achieve the same task increases exponentially with n [40]. Moreover, experimental confirmation of this assertion was provided by [41]. They proved that for some tasks, the number of experiments needed to learn a desired property from physical systems using a Quantum Recurrent Neural Network (QRNN) is exponential in n with the conventional strategy, but only polynomial in n using the quantum-enhanced strategy. Abbas et al. [42] provided numerical evidence demonstrating that a specific class of QNNs can undergo faster training, indicating an advantage for QML. This was subsequently verified on real quantum hardware. They showed that, with the use of a specific feature map, QNNs can be more trainable than classical NNs, raising high expectations for their future progress. Xiao et al. [43] presented a hybrid quantum–classical machine learning algorithm consisting of a classical Artificial Neural Network (ANN) and Parameterized Quantum Circuit (PQC) for the classification of images. They demonstrated through practical experimentation that quantum feature encoding and learning operations are hardware-efficient, making them implementable in NISQ devices. The suggested QML model maps the bucket signals into a classical neural network to reduce dimensionality. The PQC performs the clustering of features with the same class label while separating those with different labels. A linear classifier classifies the bucket signals into their respective categories. Qu et al. [44] proposed a hybrid model for diagnosis. The system consists of a pre-trained QCNN to efficiently extract features from medical images. QCNN consists of a convolutional layer, and all others are classical. The features extracted from the QCNN are subsequently combined with features from other modalities (such as blood test results or breast cell slices) and used to train an efficient Variational Quantum Classifier for intelligent diagnosis. The system achieved better accuracy than its classical counterpart. Forestano et al. [45] proved that the quantum Graph Neural Networks (QGNNs) seem to exhibit enhanced classifier performance over their classical GNN counterparts based on the best test AUC scores on an increasing size and complexity dataset from high-energy particle collisions at the CERN Large Hadron Collider (LHC). Despite this result, quantum algorithms require more time to train than classical networks.

2.2.2. Drug Discovery

Li et al. [46] proposed a QNN called the quantum Quanvolutional Neural Network (QuanNN) to classify binding pockets into one of three groups of proteins. The results showed that the QuanNN + MLP (hybrid) model outperformed single-layer CNN + MLP in terms of training and validation accuracy. In a study conducted by Li et al. [47], the researchers employed a Quantum Generative Adversarial Neural Network (QGAN) with a hybrid generator to identify novel drug molecules. The findings of the study suggest that a classical GAN cannot effectively learn molecule distribution. However, the proposed QGAN-HG, with merely 15 additional quantum gate parameters, exhibits significantly enhanced effectiveness in learning molecular tasks. Sagingalieva et al. [48] proposed a hybrid QNN for drug response prediction based on a combination of CNN, GNN, and deep quantum neural layers. The CNN learns features of cancer cell lines, GNN learns features of SMILES of drug and deep QNN predicts the drug response prediction. Compared to the classical model, the hybrid model can learn with fewer samples. Domingo et al. [49] proposed a hybrid quantum–classical 3D CNN for predicting protein–ligand binding affinity, which mitigates the complexity (the number of training parameters) compared to the classical 3D CNNs, while maintaining optimal prediction performance. Despite the quantum advantage, this model falls under complex-based input methods, relying on experimentally obtained protein–ligand complexes, thereby limiting its utility. Furthermore, 3D CNNs, compared to 1D CNNs, exhibit significantly higher complexity, demanding the learning of millions of training parameters. This reality increases the cost and the duration

of the training process. It is important to highlight that numerous algorithms have been developed with a hybrid approach in the field of drug discovery, exploiting the advantages of both classical and quantum systems [50].

Although QML has seen rapid development, there has been relatively limited research conducted on quantum regression, with most studies focusing on quantum classification. Moreover, to date, we have encountered only one study on predicting protein–ligand binding affinity using a hybrid model, and its limitations were presented in the preceding paragraph. Hence, it is crucial to develop a hybrid quantum–classical model for predicting binding affinity that is applicable in the current application environment.

2.3. Preliminaries

For a comprehensive understanding of QML, it is important to provide detailed descriptions of qubits, quantum gates, quantum algorithms, and variational quantum algorithms.

QC is a rapidly developing field that exploits the principles of quantum mechanics. Data are stored in qubits ($|0\rangle$ $|1\rangle$), which are quantum equivalents of classical bits. A logical quantum gate is a fundamental building block within a quantum circuit, similar to classical logic gates, which operate on a small set of qubits. Programs or algorithms consist of quantum gates. Examples of gates include the rotation gates RX, RY, and RZ, which rotate the state vector by an angle around the x-axis, y-axis, and z-axis on the Bloch sphere, respectively. Another example is the CNOT gate, which operates on two qubits. If the first qubit is $|1\rangle$, then the state of the second qubit will be flipped (Table 1).

Table 1. Logical quantum gates.

	RX(θ)	**RY(θ)**	**RZ(θ)**	**CNOT**
Matrix	$\begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix}$	$\begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix}$	$\begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Notation	RX(θ)	RY(θ)	RZ(θ)	

The quantum algorithm or quantum circuit is a circuit composed of numerous quantum gates, defined by three fundamental stages (Figure 1a). The initial stage converts classical data into quantum data, known as quantum embedding/encoding. The second stage involves a series of quantum gates applied to the quantum data, leading to quantum computation/evolution. Finally, measurements are performed to extract classical information from the quantum system. Variational quantum circuit (VQC) or VQA or PQC represents a class of hybrid quantum–classical optimization algorithms, where a function is assessed through quantum computation [34]. Subsequently, the parameters of this function are refined using classical optimization techniques (Figure 1b). VQC is a prominent technique within QML. Hybrid models, which combine quantum and classical machine learning methodologies, can process variational quantum circuits and derive the gradients of all observations with respect to each parameter through automatic differentiation of the circuit using a QNN.

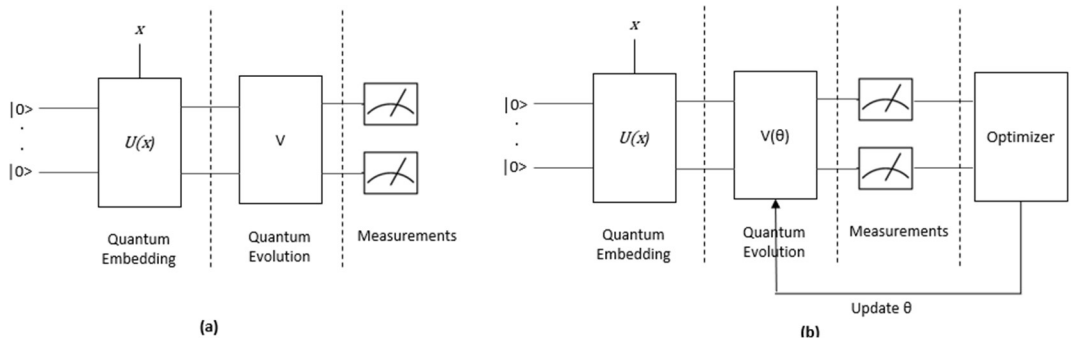

Figure 1. Architecture of (**a**) quantum algorithm/circuit; (**b**) variational quantum algorithm/circuit—(U(x) is the quantum routine of encoding classical data x to the quantum state, V is the quantum evolution, and V(θ) is the variational circuit block with trainable parameters θ) [50].

3. Materials and Methods

3.1. Overall Framework

We introduced a hybrid quantum–classical framework to predict binding affinities between ligands and protein targets. In general, our approach involved initially the implementation of an efficient classical model, followed by developing hybrid models, wherein we substituted the classical prediction module with various quantum modules and, finally, their assessment was held (Figure 2). The framework includes several stages, including data collection and preparation, input encoding, model construction, and model evaluation. In the initial stage, datasets comprising protein and ligand structures were collected for model training and performance evaluation. In the second stage, input encoding involved converting protein sequences and ligand structures (SMILES) into digital matrices to prepare the input data for the models. Each model consists of three modules: feature extraction, fusion, and prediction of binding affinity. For both proteins and ligands, the feature extraction module includes a word embedding layer followed by five 1D dilated CNN layers. In the fusion module, the outputs from the two feature extraction modules are concatenated and inputted into the subsequent module. The prediction module was implemented in two ways: first, using classical method, and second, using quantum approaches. Three classic modules were implemented, each comprising three fully connected layers of varying complexity. Additionally, five distinct quantum modules were developed, based on different quantum embedding approaches: amplitude, angle, and dense angle. Three of them used 8 qubits, while two used 4 qubits. In the fourth stage, we evaluated the predictive performance of the models using test datasets.

3.2. Datasets

We trained and evaluated classical and hybrid models on the PDBbind dataset following the methodology outlined in [13,14]. The PDBbind database [51] contains experimentally verified protein–ligand binding affinities, typically expressed as −logKi and −logKd, sourced from the Protein Data Bank [52]. Three datasets of the 2016 version of the PDBbind database and two supplementary test datasets (Test105 and Test71) sourced from the PDB were used in this study. A statistical summary of the datasets is presented in Table 2. The general, the refined set, and the core 2016 have 13,283, 4057, and 290 protein–ligand complexes, respectively. The collection of the PDBbind core set aims to offer a relatively small yet high-quality collection of protein–ligand complexes for validating docking and scoring methods. To ensure no data overlap, the PDBbind 2016 datasets underwent the same preprocessing steps as in the prior studies [13,14]. The refined set (4057 protein–ligand complexes) was subtracted from the general set (13,283 protein–ligand complexes), and the 290 protein–ligand complexes in the core 2016 set were excluded from the refined set. Moreover, to ensure a convenient model comparison, 82 protein–ligand complexes were excluded from the refined set, along with 5 protein–ligand complexes from the general set.

Hence, the final numbers of protein–target pairs were 9221, 3685, and 290 in the general set, refined set, and core 2016 set, respectively. Subsequently, 1000 protein–target complexes were randomly chosen from the refined set to form the validation set. The remaining protein–target pairs in the refined set, along with the entire general set, were combined and used as the training set. The complete core 2016 set was employed as the test dataset. Additionally, two supplementary test datasets were used in this study as in [14]. The first of these datasets, termed Test105, comprised 105 protein–ligand complexes. For each protein sequence, the Smith–Waterman similarity was ensured to be at most 60% compared to any sequence in the training set. The other test dataset, named Test71, consisted of 71 protein–ligand pairs. For each sequence, the sequence identity was kept below 35% compared to the sequences in the training dataset.

Table 2. Statistical summary of the datasets.

Dataset	Source	Protein–Ligand Complexes		Samples		
		Original	Final	Training	Validation	Test
General [51]	PDBbind	13,283	9221	9221	0	0
Refined [51]	PDBbind	4057	3685	2685	1000	0
Core 2016 [51]	PDBbind	290	290	0	0	290
Test105 [14]	PDB	105	105	0	0	105
Test71 [14]	PDB	71	71	0	0	71
Total		17,806	13,372	11,906	1000	466

We extracted the protein sequences from the protein PDB files of the datasets while gathering the SMILES strings from the ligand SDF files. Furthermore, because the lengths of SMILES and protein sequence strings vary, it is essential to enforce fixed lengths to produce an effective representation. The fixed lengths for protein sequences and SMILES strings were determined based on the distributions illustrated in Figure 3.

The total samples from all sets are 13,372 (11,906 + 1000 + 466). In many studies, the character length of SMILES strings and protein sequences is required to collectively represent at least 90% of the dataset for efficient models [14,19]. The maximum length of protein sequence is 4720 and the minimum length is 24. To cover approximately 90% of the proteins, we choose 1000 as a fixed length of protein sequence. Respectively, the maximum length of SMILES strings is 472 and the minimum is 6. To cover approximately 90% of the ligands, we choose 160 as a fixed length of SMILES string. Longer protein sequences and SMILES strings were truncated, and shorter protein sequences and SMILES strings were padded with zeros to the fixed lengths. Figure 4 shows the binding affinity distribution, which follows a normal distribution with values ranging from a minimum of 0.4 to a maximum of 15.22.

3.3. Input Representation

All models in this study contain two inputs, as shown in Figure 2: the input representations of proteins and ligands. For the ligand representations, we used the widely used 1D chemical structure, the SMILES [16], which encodes information about atoms, bonds, rings, and other molecular features. Specifically, the SMILES strings of compounds are composed of 64 characters (Supplementary Table S1). The input representation of a protein consists of amino acid sequence, which is a 1D structure. Protein sequences generally comprise 20 distinct kinds of amino acids. However, some proteins may contain non-standard residues. In such cases, we treat these non-standard residues with yet another kind of unknown amino acid. Hence, the total character sets for protein sequence representation are 21 (Supplementary Table S2).

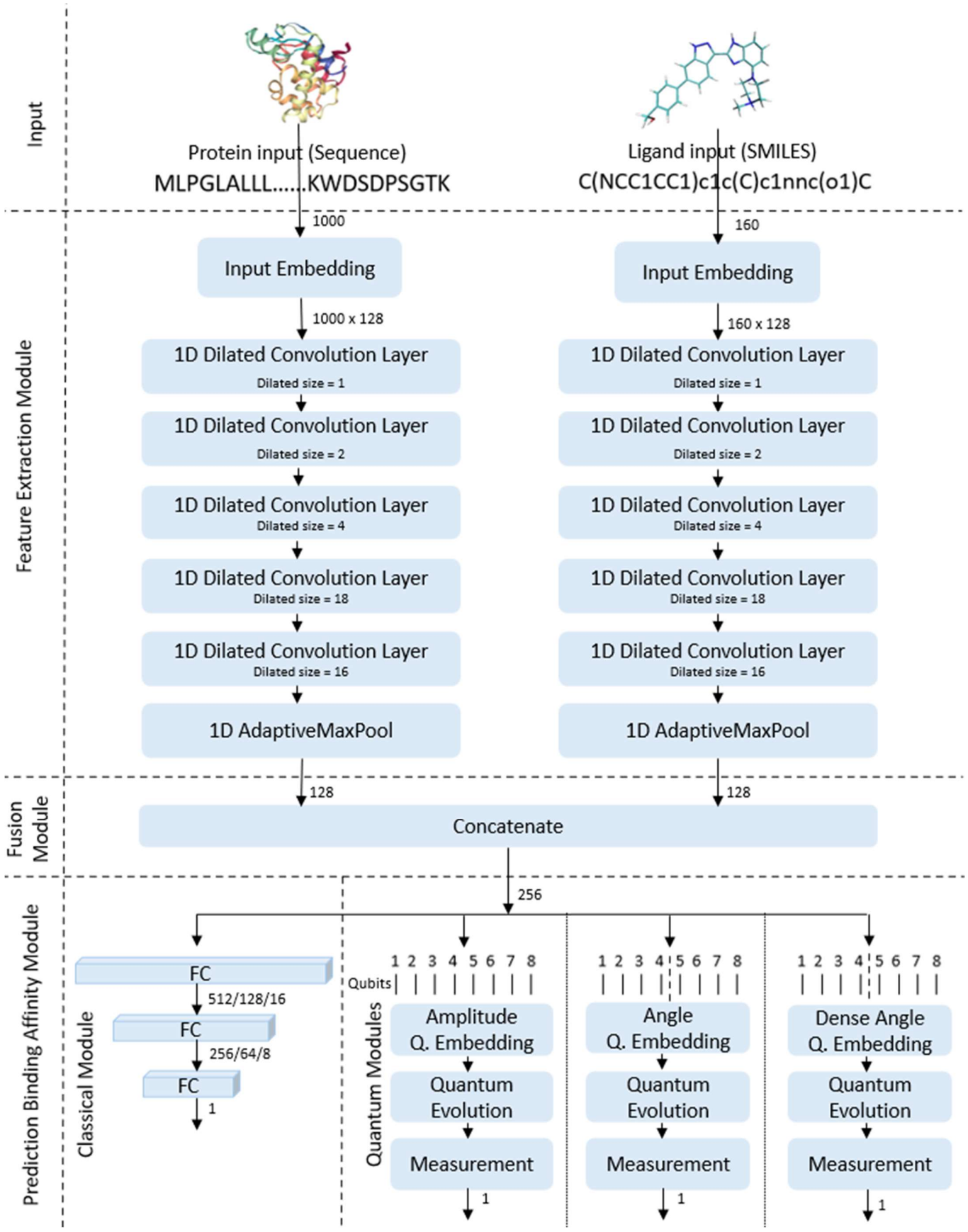

Figure 2. Overall framework architecture.

We used integer/label encoding, which employs integers to represent the categories of ligands and proteins in their inputs. Previous published works have demonstrated the effectiveness of this approach [14,23]. The integer encoding for SMILES is represented as follows, $X^D = \{x_1^D, x_2^D, \ldots x_{160}^D\} \in \mathbb{R}^{V_D}$, where V_D is the vocabulary sizes in the format of SMILES. The integer encoding for protein sequence is represented as follows, $X^S = \{x_1^S, x_2^S, \ldots x_{1000}^S\} \in \mathbb{R}^{V_S}$, where V_s is the vocabulary sizes in the format of amino acids of protein.

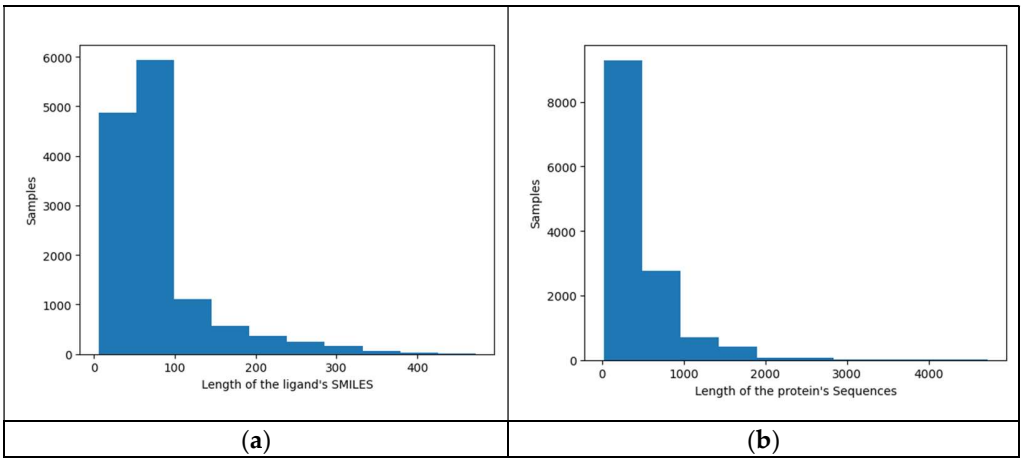

Figure 3. Distribution of lengths for (**a**) ligand SMILES; and(**b**) proteins sequences; from all datasets.

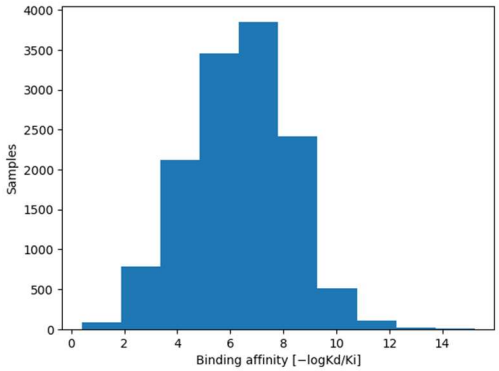

Figure 4. Distribution of binding affinity from all data.

3.4. Architecture of the Models

All models consist of an input layer, a feature extraction module, a fusion module, and a prediction module. The architecture of the models is depicted in Figure 2. The model input is detailed in the preceding Section 3.3. The feature extraction module and the prediction module are detailed as follows.

3.4.1. Feature Extraction Module

As illustrated in Figure 2, the feature extraction module includes two components: protein feature extraction and ligand feature extraction. Both parts follow the same structure, an embedding layer and five dilated convolution layers.

Embedding layer: At the embedding level, we used the word embedding technique commonly used in natural language processing (NLP) problems. That represents characters/words as dense vectors of real numbers in a continuous vector space. The size of each embedding vector chosen for embedding the SMILES protein sequences and ligands is 128. Hence, we obtain embedding matrices of sizes 1000×128 and 160×128 for each protein and ligand, respectively.

One-dimensional dilated CNN: Dilated convolution has been shown to be an effective technique for capturing large-scale multiscale intramolecular interactions in protein and ligand sequences, respectively [14,25,26]. The protein and ligand feature extraction parts contain five dilated convolution layers with a kernel size of 3, and 1, 2, 4, 8 and 16 dilation rates, respectively. Subsequently, the five dilated convolution layers are followed by an adaptive pooling layer. The output of each part is a vector of size 128.

3.4.2. Fusion Module

The outputs of protein and ligand feature extraction components are concatenated to create a vector of size 256. This vector serves as the input for both the classical and quantum modules.

3.4.3. Classical Prediction Module

In the classical approach, the prediction module consists of three fully connected (FC) linear layers. The final layer outputs the predicted binding affinity. Three classic modules of varying complexity were implemented. (256, 512, 256, 1), (256, 128, 64, 1) and (256, 16, 8, 1) are the parameters of layers of the three modules, respectively.

3.4.4. Quantum Prediction Module

We selected the VQR approach with 4 and 8 qubits to predict the binding affinity. Five quantum prediction modules were implemented based on different quantum embedding approaches. Two of them are based on angle quantum encoding, two on dense angle quantum encoding, and one on amplitude quantum encoding. To create quantum prediction modules analogous to classical ones, we used three layers of repeating quantum gates. These layers constitute the evolution part, as illustrated in Figure 5. Each layer consists of three single qubit rotations (RZ, RY, RZ) and entanglers (CNOT). In the measurement part, all qubits except the first apply a CNOT operation to the first qubit. Finally, we used the expectation value of the Pauli Z gate on the first qubit to measure the predicted binding affinity.

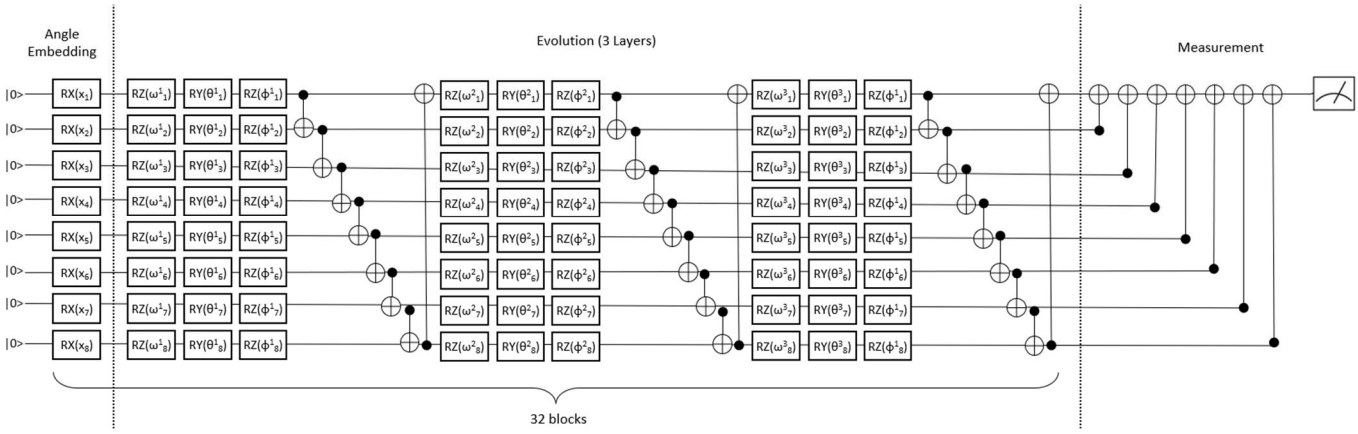

Figure 5. Prediction module (VQR) with angle embedding.

Angle Embedding: In angle encoding, represented by Equation (1), also referred to as qubit encoding or tensor product encoding, a single rotation $R(x_i)$ is applied for each feature x_i, where R—can be one of RX, RY, or RZ gate, which encodes a vector $x \, \epsilon \, \mathbb{R}^N$ into the rotation angles of n qubits, where $N \leq n$

$$|x\rangle = \otimes_{i=1}^{N} R(x_i)|0^N\rangle, \tag{1}$$

where x_i is the ith feature of x, and R can be one of RX, RY, or RZ gate.

Angle encoding uses n qubits with a quantum circuit of constant depth, making it suitable for NISQ computers. Commonly used quantum neural network architectures typically use angle encoding [36,39,42–44]. In our module, encoding 256 features requires 256 qubits. However, encoding 256 features with this approach poses challenges for today's quantum engineers for two main reasons: (a) this qubit count exceeds the capacity of fault-tolerant quantum simulators or readily accessible hardware, and (b) the noise-free barren plateau problem [53] significantly impacts this area, making it impossible to train the model. Instead, we drew inspiration from the data re-uploading method introduced in [54] and further developed in [55]. We used 8 qubits and created 32 blocks to encode 256 (8×32) features. The first eight features $\{x_i\}_{i=1}^{8}$ are encoded on the first block, the second eight features $\{x_i\}_{i=9}^{16}$ are encoded on the second block, and so on. Each block consists of angle encoding of 8 features and 3 evolution layers. In this module, we used the RX rotation gate for angle embedding. The prediction module with angle embedding is presented in Figure 5.

Dense Angle Embedding: In dense angle encoding, rather than applying a single rotation $R(x_i)$ for each feature x_i, two features are encoded on the same qubit using a combined rotation, as represented by Equation (2).

$$|x\rangle = \otimes_{i=1}^{N/2} R_a(x_{2i-1}) R_b(x_{2i}) |0^N\rangle, \tag{2}$$

where x_{2i} is the $2i$th feature of x, x_{2i-1} is the $2i-1th$ feature of x, and R_a and R_b can be one of RX, RY, or RZ. The number of qubits (n) is halved compared to the number of features (N) [56]. In the dense angle approach, we relied on the data re-uploading method, such as angle encoding. We used 8 qubits and generated 16 blocks to encode 256 features ($8 \times 16 \times 2$), with each qubit encoding 2 features. Each block consists of angle encoding of 8 features and 3 evolution layers. In this module, we used the RX and RY rotation gates for quantum embedding. Supplementary Figure S1 presents the prediction module incorporating dense angle embedding.

Amplitude Embedding: The amplitude encoding of a vector $x \, \epsilon \, \mathbb{R}^N$ is represented by Equation (3).

$$|x\rangle = \frac{1}{\| x \|_2} \sum_{i=1}^{N} x_i |i\rangle, \tag{3}$$

where x_i is the ith feature of x, $|i\rangle$ is the ith computational basis state, and $\| x \|_2$ is the 2-norm of x. In amplitude embedding, a quantum state comprising $n = \log_2 N$ qubits can represent a data point with N features. Quantum amplitude prediction module encoded the 256 features using 8 qubits. The prediction module with amplitude embedding is presented in Supplementary Figures S2 and S3.

3.5. Evaluation Metrics

In this study, six performance metrics were employed to evaluate and compare the predictive performance of classical and hybrid models. These metrics included the Concordance Index (CI), Mean Squared Error (MSE), Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Pearson correlation coefficient (R), and Standard Deviation (SD) in regression.

CI, represented by Equation (4), is derived from the probability between the predicted values and the actual values for two randomly selected protein–ligand complexes in a specified order. It serves to evaluate the model's degree of fit. The CI value ranges from 0 to 1, with higher values indicating better prediction performance.

$$CI = \frac{1}{Z} \sum_{t_i > t_j} h(p_i - p_j), \tag{4}$$

where p_i is the predicted value for the larger actual binding affinity value t_i and p_j is the predicted value for the smaller actual affinity value t_j. The normalization constant Z is

the total number of protein–ligand complexes, and the step function $h(x)$ is defined as in Equation (5).

$$h(x) = \begin{cases} 1 & x > 0 \\ 0.5 & x = 0 \\ 0 & x < 0 \end{cases}, \tag{5}$$

The MSE, represented by Equation (6), value evaluates the prediction accuracy of the model, representing the difference between the predicted values and the actual values. Lower MSE values indicate a better-performing model. MSE is differentiable, implying the existence of a derivative at every point, enabling it to be used as a loss function.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(t_i - p_i)^2, \tag{6}$$

where n is the number of protein–ligand complexes, t_i refers to the actual binding affinity of the sample indexed with i, and p_i refers to the predicted binding affinity of the sample indexed with i. RMSE is the square root of MSE and is also used as loss function.

The MAE, represented by Equation (7), is calculated as the sum of all the differences between the actual and predicted values divided by the total number of samples in the dataset. In other words MAE is the absolute average distance of model prediction. Lower MAE values indicate a better-performing model. This metric is also used as the metric of prediction error and is calculated as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|t_i - p_i|, \tag{7}$$

where n is the number of protein–ligand complexes, t_i refers to the actual binding affinity of the sample indexed with i, and p_i refers to the predicted binding affinity of the sample indexed with i.

The Pearson correlation coefficient, represented by Equation (8), is a measure that calculates the linear correlation between the predicted value p_i and the actual value t_i.

$$R = \frac{\sum_{i=1}^{n}(t_i - \overline{t})(p_i - \overline{p})}{\sqrt{\sum_{i=1}^{n}(t_i - \overline{t})}\sqrt{\sum_{i=1}^{n}(p_i - \overline{p})}}, \tag{8}$$

where n is the number of protein–ligand complexes, t_i refers to the actual binding affinity of the sample indexed with i, p_i refers to the predicted binding affinity of the sample indexed with i, $\overline{t}$ refers to the mean of the actual binding affinity of n samples, and $\overline{p}$ refers to the mean of the predicted binding affinity of n samples. The range of R values is between -1 and +1. A value of +1 indicates perfect positive correlation between actual and predicted values, while -1 indicates perfect negative correlation. A value of 0 indicates no correlation between the experimental and predicted values.

The SD in regression quantifies the degree of imprecision [57] and is determined as in Equation (9).

$$SD = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}[t_i - (a * p_i + b)]^2}, \tag{9}$$

where n is the number of protein–ligand complexes, t_i and p_i are the actual and predicted binding affinities of the sample i, respectively. a and b are slope and intercept of the function line between actual and predicted values.

4. Experiment—Results

For the experiment, we followed the framework depicted in Figure 2. Initially, we deployed various classical models using PyTorch [58] for predicting binding affinity and subsequently assessed their performance to identify the most effective one, aiming to choose

the optimal feature extraction module. Then, we developed hybrid quantum–classical models using PennyLane [59], wherein the classical prediction module was replaced with quantum. Following this, an evaluation was conducted. For training, validation, and testing all models, we used the respective datasets outlined in Section 3.2. All machine learning was carried out in a quantum simulator. The specifications of the machine used for conducting the experiments and the versions of the packages employed are detailed in Supplementary Table S3.

4.1. Normalization in Regression Prediction Module

As discussed in Section 3.4.4, for evaluating the output of the quantum prediction module, we used the expectation value of the Pauli Z gate, which yields potential output values within the range of $[-1, +1]$. On the other hand, as mentioned in Section 3.2, the binding affinity values fall within the interval [0.4, 15.22]. Therefore, preprocessing to normalize the output variable within the interval $[-1, +1]$ is required to ensure proper training of the quantum regression module. Since binding affinity values are positive, normalization to the interval [0, 1] is selected. In conclusion, to compare a quantum regression module with any classical counterpart, it is imperative that the prediction values be normalized to the output range of the quantum regression module.

Figure 6 illustrates three different normalization techniques using the scikit-learn [60] package's MinMaxScaler, StandardScaler, and RobustScaler. These functions were executed using the following parameters: MinMaxScaler (feature_range = (0, 1)), StandardScaler (with_mean = False, with_std = True), and RobustScaler (quantile_range = (25.0, 75.0)). MinMaxScaler scaled and transformed the output values to be between zero and one. StandardScaler transformed the output values to have a standard deviation of one. RobustScaler scaled the output values according to the quantile range between the 1st quartile (25th percentile) and the 3rd quartile (75th percentile). MinMaxScaler normalized the values to the range [0, 1], while StandardScaler and RobustScaler normalized the values to the range [0.2, 8.0] and $[-2.2, 3.2]$, respectively. Therefore, the MinMaxScaler function was employed in our experiment.

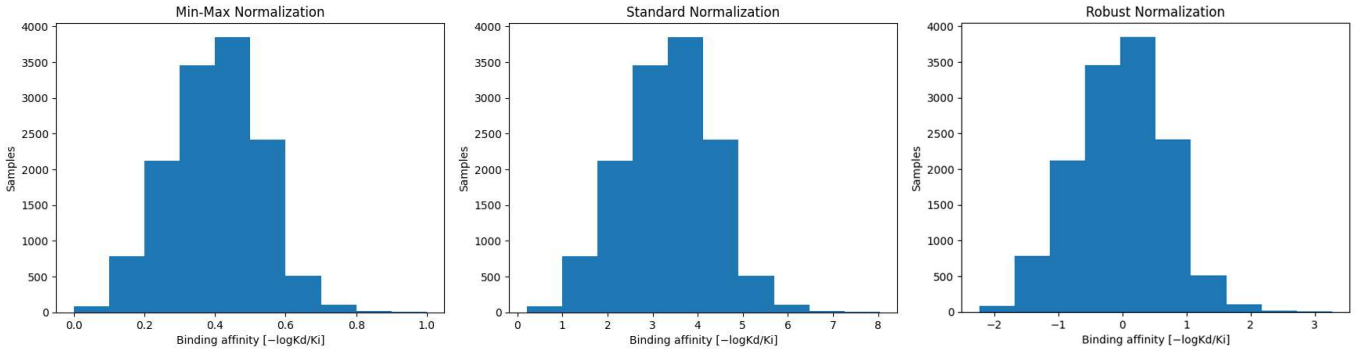

Figure 6. MinMax, standard, and robust normalization.

4.2. Model Training

In all classical models, both convolutional and FC layers included the rectified linear unit (ReLU) activation function, as defined by the formula presented in Equation (10).

$$R(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}, \tag{10}$$

In the hybrid model, the last ReLU activation function in the feature extraction module was replaced by the Sigmoid activation function. The sigmoid function, represented by Equation (11), maps the incoming inputs to a range between 0 and 1

$$R(x) = \frac{1}{1 + e^{-x}}, \tag{11}$$

The output in a range of 0 to 1 is multiplied by π so that the input to the quantum regression module falls within a range of 0 to π. The crucial factor was the selection of the RX gate for the initial gate in angle and dense embedding. The RX gate returns distinct values when the rotation angle falls within the interval 0 to π, while it can yield identical values for different inputs within a more extended interval.

Additionally, to minimize the loss function, we optimized the model parameters using the AdamW optimizer [61] with a maximum learning rate of 0.0001. We employed a weight decay of 0.01 to update the model's weights. We used the MSE loss function, which creates a criterion to measure the mean squared error, thereby minimizing the difference between the actual and prediction value during training. To optimize the parameters and define our model, we used a batch size of 256 and trained for 30 epochs. Subsequently, we selected the model with the lowest error on the validation set. The parameters for all experiments are listed in Supplementary Table S4.

4.3. Choose Classical Feature Extraction Module

The initial objective was to select the classical feature extraction module with the highest efficiency. We compared the feature extraction performance of five classical models, as described in Section 3.4.1. These five models (C1-DTA, C2-DTA, C3-DTA, C4-DTA, C5-DTA) were developed using varying numbers of 1D dilated convolution layers—ranging from one to five layers—and dilation sizes of 1, 2, 4, 8, and 16, respectively.

The model's accuracy is represented by CI, R, MSE, SD, MAE, and RSME in the validation dataset. According to the comparison results presented in Figure 7, we conclude that the model (C5-DTA) using five dilated convolution layers and dilation sizes of 1, 2, 4, 8, and 16, respectively, in the feature extraction module exhibits notable performance in terms of all metrics for feature extractions. The C5-DTA model exhibits superior CI, R, and lower MSE, RMSE, MAE, and SD values for the validation dataset. Examining the MSE diagram, which also serves as the loss function, we observe that the C1-DTA model is not learning in a stable and consistent manner.

While analyzing the training dataset evaluation (Supplementary Figure S4), although C4-DTA outperformed the other models, we selected the model with the lowest error on the validation dataset, which was C5-DTA. Hence, the feature extraction module of C5-DTA was selected to proceed with the experiment.

4.4. Hybrid Quantum–Classical Models

Subsequently, we implemented five hybrid quantum–classical models by substituting the classical module of the prediction with a quantum. Meanwhile, we retained the classical 1D CNNs (five 1D dilated convolution layers) for feature extractions, as assessed in the previous experiment (Section 4.3). The only difference lies in the last activation function; ReLU was replaced with sigmoid (Section 4.2). The primary distinction of the hybrid models lies in the quantum embedding approach (Section 3.4.4). The three quantum embedding approaches employed were amplitude (HQ-DTA-amQE-8), angle (HQ-DTA-anQE-8), and dense angle (HQ-DTA-danQE-8), each using 8 qubits. Furthermore, considering the crucial role of depth and number of gates in the quantum circuit, we introduced two additional quantum prediction models using angle (HQ-DTA-anQE-4) and dense angle (HQ-DTA-danQE-4) quantum embeddings, each employing 4 qubits. Notably, the implementation of amplitude quantum embedding was blocked due to the limitation of 4 qubits.

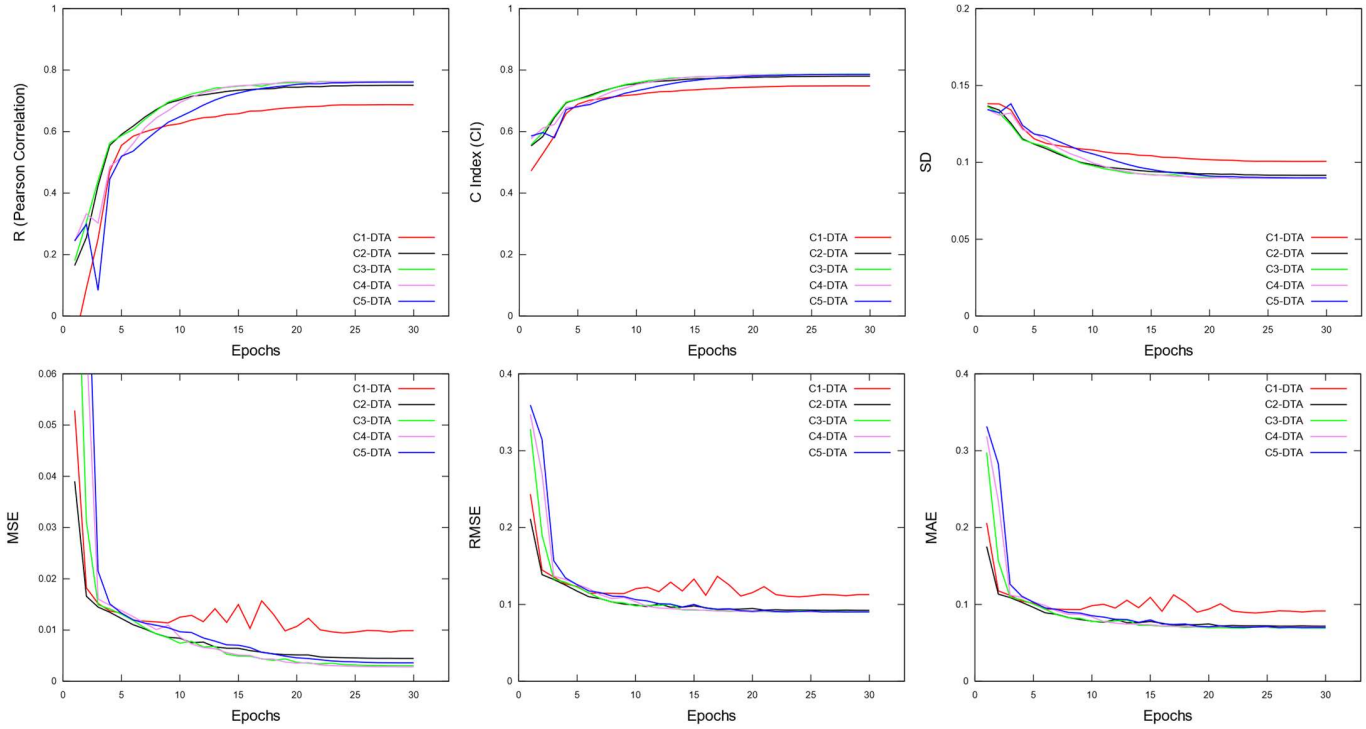

Figure 7. Evaluation metrics of C1-DTA, C2-DTA, C3-DTA, C4-DTA and C5-DTA in validation dataset.

4.5. Classical Models

The classical model C5-DTA and all quantum prediction modules differ significantly in complexity, a contrast reflected in the quantity of trainable parameters they have (Table 3). The classical prediction module of C5-DTA has hundreds of thousands of trainable parameters (263,169), whereas the quantum modules of hybrid model have trainable parameters from dozens to a few thousand (72 to 2304). For a more robust assessment of classical versus quantum machine learning, we incorporated two extra classical prediction modules with fewer parameters and three FC layers (Section 3.4.3). The C5-DTA-mc (256, 128, 64, 1) and C5-DTA-lc (256, 16, 8, 1) have 41,217 and 4256 trainable parameters, respectively. Despite its higher complexity, the second module approaches the complexity level of quantum prediction modules.

Table 3. Trainable parameters of classical and quantum prediction modules.

Model	Layers	Qubits	Quantum Embedding	Formula of Trainable Parameters of Prediction Module	Trainable Parameters of Prediction Module
C5-DTA	256, 512, 256, 1			(in_1 × out_1 + × out_1) + (in_2 × out_2 + × out_2) + (in_3 × out_3 + × out_3)	(256 × 512 + 512) + (512 × 256 + 256) + (256 × 1 + 1) = 263,169
C5-DTA-mc	256, 128, 64, 1				(256 × 128 + 128) + (128 × 64 + 64) + (64 × 1 + 1) = 41,217
C5-DTA-lc	256, 16, 8, 1				(256 × 16 + 16) + (16 × 8 + 8) + (8 × 1 + 1) = 4256
HQ-DTA-amQE-8		8	Amplitude	layers × qubits × rotation	3 × 8 × 3 = 72
HQ-DTA-anQE-8		8	Angle	blocks × layers × qubits × rotation	32 × 3 × 8 × 3 = 2304
HQ-DTA-danQE-8		8	Dense Angle		16 × 3 × 8 × 3 = 1152
HQ-DTA-anQE-4		4	Angle		64 × 3 × 4 × 3 = 2304
HQ-DTA-danQE-4		4	Dense Angle		32 × 3 × 4 × 3 = 1152

4.6. Complexity

Table 3 lists the training parameters for the classical and quantum regression prediction modules.

All modules maintain consistency in the number of layers, set at three. The in_x and out_x represent the input and output features, respectively, of layer x, where x ranges from 1 to 3. The qubit rotations involve three parameters: the rotation angles ω, θ and φ (Figure 5). The variable "blocks" denotes the number of blocks required to encode 256 features, 32 for angle encoding and 16 for dense angle encoding for the circuit with 8 qubits and 64 for angle encoding and 32 for dense angle encoding for the circuit with 4 qubits.

For the classical models, there exists a parameter scaling ranging from high (C5-DTA) and medium (C5-DTA-mc) to low (C5-DTA-lw) with 263,169, 41,217, and 4256 trainable parameters, respectively. C5-DTA-lw can be regarded as comparable in complexity to quantum modules. The HQ-DTA-anQE-8 and HQ-DTA-anQE-4 have identical parameter counts (2304), surpassing the other quantum modules in parameters, while HQ-DTA-danQE-8 and HQ-DTA-danQE-4 have half that number (1152). Finally, HQ-DTA-amQE-8 has the fewest parameters (72).

4.7. Gates and Depth of Quantum Modules

The analysis of the quantum regression prediction module, considering the number of gates and circuit depth, is detailed in Table 4. This enables conclusions to be drawn regarding the performance of models.

Table 4. Gates and circuit depth of quantum regression prediction modules (red color: quantum embedding, green color: evolution, blue color: measurement).

Model	Blocks	Layers	RX	RY	RZ	CNOT	Total Gates	Depth
HQ-DTA-amQE-8	1	3		255 + 24	255 + 24 × 2	508 + 24 + 7	1121	1004 + (24 + 3 × 3) × 1+ 7 = 1044
HQ-DTA-anQE-8	32	3	256	24 × 32	2 × 24 × 32	24 × 32 + 7	3335	32 + (24 + 3 × 3) × 32 + 7 = 1095
HQ-DTA-danQE-8	16	3	128	128 + 24 × 16	2 × 24 × 16	24 × 16 + 7	1799	16 × 2 + (24 + 3 × 3) × 16 + 7 = 567
HQ-DTA-anQE-4	64	3	256	12 × 64	2 × 12 × 64	12 × 64 + 7	3335	64 + (12 + 3 × 3) × 64 + 7 = 1223
HQ-DTA-danQE-4	32	3	128	128 + 12 × 32	2 × 12 × 32	12 × 32 + 7	1799	32 × 2 + (12 + 3 × 3) × 32 + 7 = 743

As anticipated, quantum modules with identical encoding exhibit the same number of gates. The module based on angle quantum embedding has the highest count (3335), while the one based on dense angle possesses about half (1799), and the amplitude-based module has the fewest (1121). This is due to the fact that in angle embedding, one qubit encodes one feature, whereas in dense embedding, it encodes two features. When considering depth, the situation is reversed for angle and dense angle embedding. In amplitude embedding, the majority of gates and the greatest depth are associated with encoding rather than evolution.

4.8. Evaluation on Training and Validation Datasets

We evaluated the classical models (C5-DTA, C5-DTA-mc, C5-DTA-lc) and the hybrid models (HQ-DTA-amQE-8, HQ-DTA-anQE-8, HQ-DTA-danQE-8, HQ-DTA-anQE-4, HQ-DTA-danQE-4). Figure 8 illustrates the evolution of the loss function (MSE) throughout the training process for both the training and validation datasets.

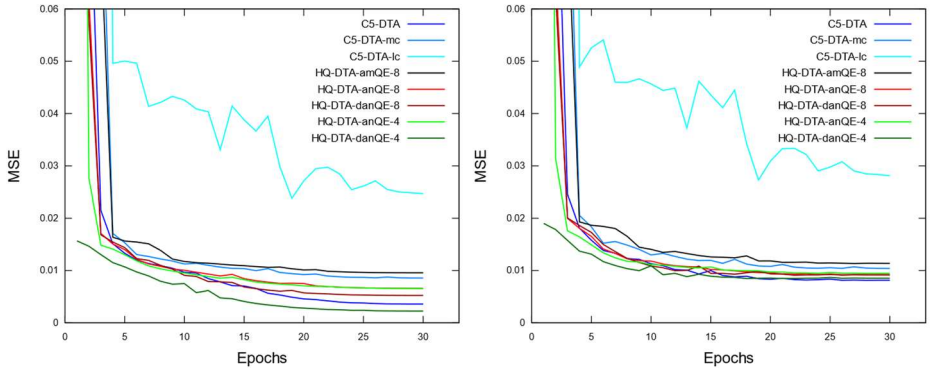

Figure 8. MSE loss function for classical (C5-DTA, C5-DTA-mc, C5-DTA-lc) and hybrid (HQ-DTA-amQE-8, HQ-DTA-anQE-8, HQ-DTA-danQE-8, HQ-DTA-anQE-4, HQ-DTA-danQE-4) models on training and validation datasets.

We observe that the MSE loss function of hybrid HQ-DTA-danQE-4 outperforms all the other models on the training dataset, while the equivalent in complexity, the classical model (C5-DTA-lc), exhibits volatile behavior. In the validation dataset, used for selecting the optimal hyperparameters of the model, the classical C5-DTA shows that it slightly outperforms the other models, whereas the C5-DTA-lc consistently exhibits the poorest performance. The MSE values show a greater deviation in the training dataset compared to the validation dataset.

From the experimental findings presented in Figure 9, it can be concluded that the C5-DTA model exhibits superior CI, R, and lower MSE, RMSE, MAE, and SD values for the validation dataset. This is due to its substantial number of trainable parameters (263,169). It is closely followed by the hybrid HQ-DTA-danQE-4 with 1152 trainable parameters, which demonstrates speed learning in terms of epochs and faster stabilization compared to the other models. Hybrid models based on angle and dense angle quantum embedding perform better than C5-DTA-mc, with the latter having 1689% and 3478% more trainable parameters, respectively. In the penultimate position is the hybrid HQ-DTA-amQE-8, with the quantum equivalent C5-DTA-lc ranking last. Therefore, comparing classical and quantum models with approximately the same complexity (in terms of the number of trainable parameters), we conclude that quantum models significantly outperform classical ones.

Among the hybrid models, the one based on dense angle quantum embedding exhibits superiority in speed learning and faster stabilization in comparison. This suggests that the dense angle quantum embedding is well suited for regression prediction tasks followed by angle embedding. Finally, we recommend avoiding the use of amplitude embedding. One of the primary advantages of dense angle embedding is its ability to be executed in constant time with parallelism; each qubit goes through two rotation gates simultaneously. Quantum systems are liable to errors arising from noise and decoherence. Table 4 reveals that the quantum circuit employing dense angle (HQ-DTA-danQE-8, HQ-DTA-danQE-4) embedding exhibits the shallowest depth and the fewest gates. Additionally, these models have the smallest number of trainable parameters (Table 3), excluding HQ-DTA-amQE-8. This configuration decreases the probability of error accumulation, enhancing the reliability of the circuits. Moreover, shallow circuits like HQ-DTA-danQE-4 become more cost-effective for NISQ devices, which face challenges related to scalability and coherence time and for quantum simulators, which integrate few qubits.

The HQ-DTA-danQE-4 outperforms all other models across all metrics on the training dataset (Supplementary Figure S5), followed by C5-DTA, while the classical analogue in complexity (C5-DTA-lc) has the lowest performance.

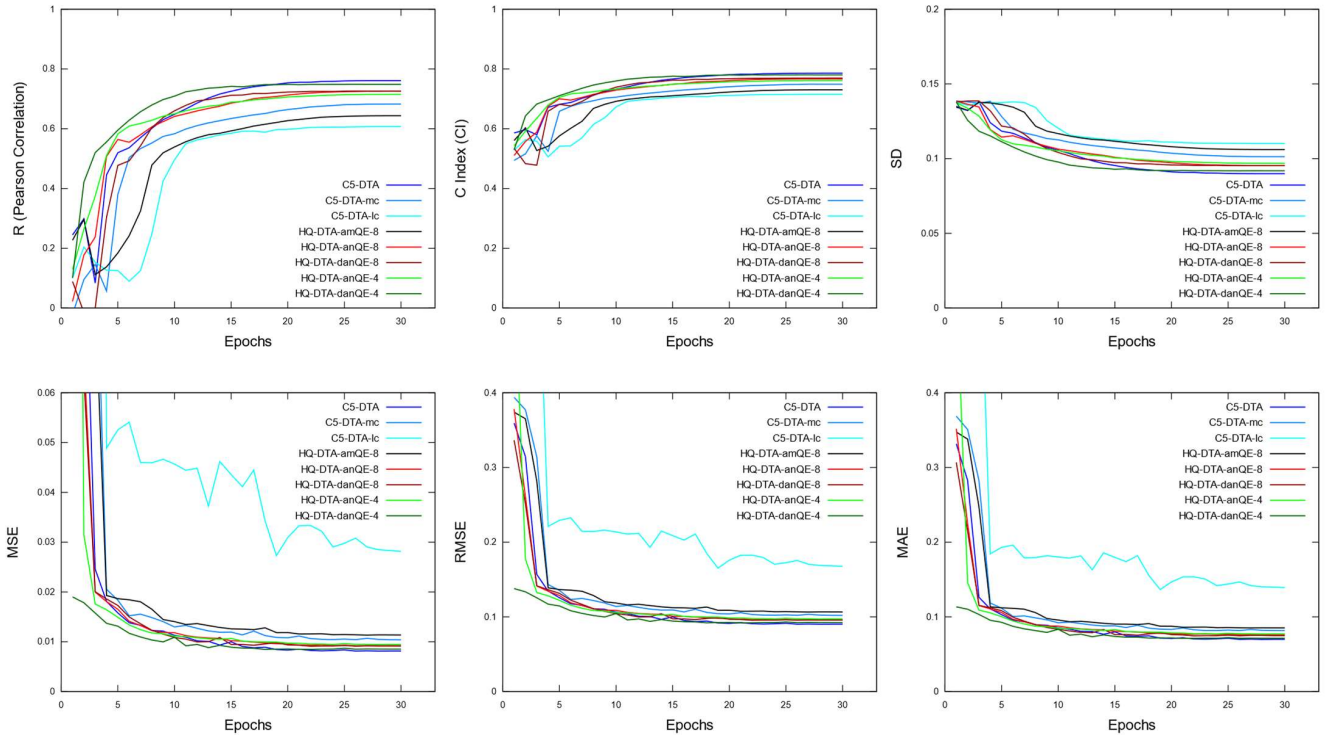

Figure 9. Metrics for classical (C5-DTA, C5-DTA-mc, C5-DTA-lc) and hybrid (HQ-DTA-amQE-8, HQ-DTA-anQE-8, HQ-DTA-danQE-8, HQ-DTA-anQE-4, HQ-DTA-danQE-4) models on validation dataset.

4.9. Evaluation on Test Datasets

Comparisons of the Core 2016 test dataset, as well as the independent Test105 and Test71 datasets, are presented in Table 5, Table 6, and Table 7, respectively.

Table 5. Metrics for core 2016 dataset (bold is the best value, ↓ lower value is better, ↑ higher value is better).

Model	Parameters of Prediction Module	MSE (↓)	RMSE (↓)	MAE (↓)	R (↑)	CI (↑)	SD (↓)
C5-DTA	263,169	0.0092	0.0963	0.0790	0.7605	0.7792	0.0954
C5-DTA-mc	41,217	0.0119	0.1094	0.0882	0.6995	0.7532	0.1049
C5-DTA-lc	4256	0.0301	0.1737	0.1420	0.5927	0.7088	0.1183
HQ-DTA-amQE-8	72	0.0130	0.1138	0.0917	0.6511	0.7298	0.1115
HQ-DTA-anQE-8	2304	0.0105	0.1026	0.0822	0.7344	0.7690	0.0997
HQ-DTA-danQE-8	1152	0.0100	0.1003	0.0809	0.7367	0.7690	0.0992
HQ-DTA-anQE-4	2304	0.0108	0.1043	0.0843	0.7291	0.7664	0.1005
HQ-DTA-danQE-4	1152	**0.0090**	**0.0953**	**0.0781**	**0.7699**	**0.7829**	**0.0937**

Hybrid HQ-DTA-danQE-4 exhibited the best performance on the test core PDBind 2016 than all models in all metrics. Particularly, HQ-DTA-danQE-4 demonstrates an enhancement in terms of R, with a value of (0.7699). Quantum superiority is observed not only in model complexity but also in accuracy. The classical C5-DTA model, which has a substantial number of trainable parameters, ranked second with an R value of 0.7605, followed by HQ-DTA-danQE-8 with an R value of 0.7367, HQ-DTA-anQE-8 with an R value of 0.7344, the HQ-DTA-anQE-4 (0.7291), the C5-DTA-mc (0.6995), HQ-DTA-amQE (0.6511) and, last, the C5-DTA-lc (0.5927). The ranking of the models remains the same across the remaining metrics. All hybrid models exceed the classical equivalent in complexity (C5-DTA-lc).

Table 6. Metrics for Test105 dataset (bold is the best value, ↓ lower value is better, ↑ higher value is better).

Model	Parameters of Prediction Module	MSE (↓)	RMSE (↓)	MAE (↓)	R (↑)	CI (↑)	SD (↓)
C5-DTA	263,169	0.0116	0.1077	0.0888	0.6075	0.7210	0.0957
C5-DTA-mc	41,217	0.0108	0.1041	0.0852	0.6092	0.7209	0.0955
C5-DTA-lc	4256	0.0132	0.1151	0.0906	0.6077	0.7177	0.0957
HQ-DTA-amQE-8	72	0.0114	0.1068	0.0854	0.6168	0.7180	0.0949
HQ-DTA-anQE-8	2304	**0.0106**	**0.1028**	**0.0851**	**0.6275**	**0.7219**	**0.0939**
HQ-DTA-danQE-8	1152	0.0115	0.1073	0.0890	0.5838	0.7100	0.0978
HQ-DTA-anQE-4	2304	0.0109	0.1045	0.0870	0.5984	0.7172	0.0965
HQ-DTA-danQE-4	1152	0.0119	0.1092	0.0897	0.5999	0.7146	0.0964

First, it can be observed that all models achieve substantially worse values on metrics in Test105 compared to those in the PDBind core 2016 test set. In the independent Test105, another hybrid model (HQ-DTA-anQE-8) exhibits the highest performance, showing quantum superiority in terms of complexity, accuracy, and generalization. Specifically, HQ-DTA-anQE-8 achieves R values of 0.6275, following the HQ-DTA-amQE with an R value of 0.6168. The C5-DTA-mc ranked third with an R value of 0.6092, followed by C5-DTA-lc with an R value of 0.6077. We conclude that encoding a feature into a qubit enhances generalization.

Table 7. Metrics for Test71 dataset (bold is the best value, ↓ lower value is better, ↑ higher value is better).

Model	Parameters of Prediction Module	MSE (↓)	RMSE (↓)	MAE (↓)	R (↑)	CI (↑)	SD (↓)
C5-DTA	263,169	0.0066	0.0813	0.0684	0.4869	0.6600	0.0801
C5-DTA-mc	41,217	0.0065	0.0810	0.0684	0.4796	0.6532	0.0805
C5-DTA-lc	4256	0.0138	0.1178	0.0930	0.3625	0.6205	0.0855
HQ-DTA-amQE-8	72	0.0076	0.0874	0.0746	0.3984	0.6149	0.0842
HQ-DTA-anQE-8	2304	0.0066	0.0810	0.0676	0.4788	0.6544	0.0806
HQ-DTA-danQE-8	1152	0.0088	0.0938	0.0815	0.2900	0.5861	0.0878
HQ-DTA-anQE-4	2304	**0.0062**	**0.0790**	**0.0653**	**0.5143**	**0.6694**	**0.0787**
HQ-DTA-danQE-4	1152	0.0069	0.0831	0.0661	0.4678	0.6431	0.0811

First, it can be observed that all models achieve substantially worse values on metrics in Test71 compared to those in the TEST105 test set. The HQ-DTA-anQE-4 outperforms all other models across all metrics, followed by all other models.

These comparison results suggest that the hybrid models exhibit significantly superior accuracy, complexity, and generalization capabilities compared to the classical counterpart on various test datasets. In summary, the benchmark test dataset and the independent source tests demonstrate the superiority of hybrid models over the classical counterpart model in binding affinity prediction.

4.10. Execution Time

According to Table 8, it is evident that the classical models have the shortest execution time for 30 epochs. While quantum prediction modules have fewer training parameters compared to their classical counterparts, the greater execution time is a result of the experimental environment, using a quantum simulator instead of a real quantum device. Among the quantum modules, HQ-DTA-amQE-8 shows the fastest execution time, and this is due to the minimal number of parameters that the model needs to learn and the reduced gradient computations. The remaining hybrids follow in terms of execution time.

Table 8. Execution time (training, validation, and testing).

Model	Time	Epoch of Best Model
C5-DTA	3:36:51.821851	28
C5-DTA-mc	3:37:51.861312	30
C5-DTA-lc	3:38:16.859586	19
HQ-DTA-amQE-8	3:42:06.893518	27
HQ-DTA-anQE-8	5:19:17.879674	28
HQ-DTA-danQE-8	4:33:20.657034	22
HQ-DTA-anQE-4	4:37:57.012041	28
HQ-DTA-danQE-4	5:47:38.876147	18

5. Conclusions and Future Work

Accurately predicting the binding affinity of a drug candidate can help the design and optimization of new molecules that bind to a target protein with better affinity (hit identification and optimization) as well as identifying new uses for approved or investigational drugs that are outside the scope of the original medical indication (drug repositioning). The drug repositioning technique is a less expensive and time-consuming alternative to traditional drug discovery methods. Recently, it has garnered significant interest from researchers due to the availability of millions of existing molecules. Deep learning methods provide promising results in this area, as they can simultaneously learn the features of ligands and proteins and subsequently predict their binding affinity. Unfortunately, two of the biggest challenges of deep learning approaches are their high complexity, as they require learning millions of training parameters, and the long training time. QC and QML theoretically offer promising solutions to effectively address these challenges.

This study proposes a hybrid quantum–classical framework to predict protein–ligand binding affinity. Initially, CNNs are selected for extracting and learning the features of ligands and proteins. Then, retaining the classical module for feature extraction, we implement various quantum and classical modules for binding affinity prediction, which accept the concatenated features as input. Quantum predicted modules are implemented with VQRs and are based on different quantum embedding approaches, while classical predicted modules are implemented with fully connected layers with different input and output parameters. The experimental results demonstrate that hybrid quantum–classical machine learning methods accelerate the training process in terms of epochs and achieve faster stabilization. The hybrid models consistently achieve superior accuracy across all conducted tests. Also, all the hybrid models significantly outperform the classical ones in accuracy with approximately the same complexity. Furthermore, hybrid models demonstrate significantly superior generalization capabilities compared to classical models. Among the hybrid models, the one based on dense angle quantum embedding demonstrates superiority in terms of speed of training and faster stabilization, whereas the use of amplitude embedding is not recommended. Overall, the hybrid quantum–classical models demonstrate quantum superiority in terms of complexity, accuracy, and generalization, thereby indicating a promising direction for QML.

Running quantum algorithms on a quantum simulator is a limitation. Future research should reconfirm these findings by running quantum algorithms on real, available quantum devices. However, the availability of many quantum simulators facilitates the implementation of quantum algorithms, while the long wait times and costs of running them on real quantum computers pose a significant obstacle for researchers.

Future work will focus on implementing quantum convolutional networks for feature extraction, as well as incorporating additional features for ligands (such as physicochemical properties) and proteins (such as secondary structure elements).

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/math12152372/s1, Supplementary Table S1: Smiles Character Set; Supplementary Table S2: Protein Sequence Character Set; Supplementary Figure S1: Prediction module (VQR) with dense angle embedding; Supplementary Figure S2: Prediction module (VQR) with amplitude embedding; Supplementary Figure S3: Amplitude embedding with 4 qubits; Supplementary Table S3: The experimental environment; Supplementary Table S4: The parameters for all experiments; Supplementary Figure S4: Evaluation metrics of classical C1-DTA, C2-DTA, C3-DTA, C4-DTA and C5-DTA on training dataset; Supplementary Figure S5: Evaluation metrics of classical (C5-DTA, C5-DTA-mc, C5-DTA-lc) and hybrid (HQ-DTA-amQE-8, HQ-DTA-anQE-8, HQ-DTA-danQE-8, HQ-DTA-anQE-4, HQ-DTA-danQE-4) models on training dataset.

Author Contributions: Conceptualization, M.A., I.K.S. and A.V.; methodology, M.A.; software, M.A.; validation, M.A., A.V. and A.T.; resources, M.A.; writing—original draft preparation, M.A.; writing—review and editing, M.A., A.V. and G.G.; visualization, M.A. and G.G.; supervision, M.A., I.K.S., A.V., A.T. and G.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The software source code and data are openly available in GitHub at https://github.com/mavramouli/HQNN-BindingAffinity (accessed on 20 June 2024).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Carracedo-Reboredo, P.; Liñares-Blanco, J.; Rodríguez-Fernández, N.; Cedrón, F.; Novoa, F.J.; Carballal, A.; Maojo, V.; Pazos, A.; Fernandez-Lozano, C. A review on machine learning approaches and trends in drug discovery. *Comput. Struct. Biotechnol. J.* **2021**, *19*, 4538–4558. [CrossRef] [PubMed]
2. Ashburn, T.T.; Thor, K.B. Drug repositioning: Identifying and developing new uses for existing drugs. *Nat. Rev. Drug Discov.* **2004**, *3*, 673–683. [CrossRef] [PubMed]
3. Mofidifar, S.; Sohraby, F.; Bagheri, M.; Aryapour, H. Repurposing existing drugs for new AMPK activators as a strategy to extend lifespan: A computer-aided drug discovery study. *Biogerontology* **2018**, *19*, 133–143. [CrossRef] [PubMed]
4. Wang, D.D.; Ou-Yang, L.; Xie, H.; Zhu, M.; Yan, H. Predicting the impacts of mutations on protein-ligand binding affinity based on molecular dynamics simulations and machine learning methods. *Comput. Struct. Biotechnol. J.* **2020**, *18*, 439–454. [CrossRef] [PubMed]
5. Wang, D.D.; Zhu, M.; Yan, H. Computationally predicting binding affinity in protein–ligand complexes: Free energy-based simulations and machine learning-based scoring functions. *Brief. Bioinform.* **2021**, *22*, bbaa107. [CrossRef] [PubMed]
6. Ballester, P.J.; Mitchell, J.B. A machine learning approach to predicting protein–ligand binding affinity with applications to molecular docking. *Bioinformatics* **2010**, *26*, 1169–1175. [CrossRef] [PubMed]
7. Shar, P.A.; Tao, W.; Gao, S.; Huang, C.; Li, B.; Zhang, W.; Shahen, M.; Zheng, C.; Wang, Y. Pred-binding: Large-scale protein–ligand binding affinity prediction. *J. Enzym. Inhib. Med. Chem.* **2016**, *31*, 1443–1450. [CrossRef] [PubMed]
8. Chauhan, N.K.; Singh, K. A review on conventional machine learning vs deep learning. In Proceedings of the 2018 International Conference on Computing, Power and Communication Technologies (GUCON), Greater Noida, India, 28–29 September 2018.
9. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
10. Biamonte, J.; Wittek, P.; Pancotti, N.; Rebentrost, P.; Wiebe, N.; Lloyd, S. Quantum machine learning. *Nature* **2017**, *549*, 195–202. [CrossRef]
11. Yu, J.L.; Dai, Q.Q.; Li, G.B. Deep learning in target prediction and drug repositioning: Recent advances and challenges. *Drug Discov. Today* **2022**, *27*, 1796–1814. [CrossRef]
12. Cang, Z.; Wei, G.W. TopologyNet: Topology based deep convolutional and multi-task neural networks for biomolecular property predictions. *PLoS Comput. Biol.* **2017**, *13*, e1005690. [CrossRef] [PubMed]
13. Stepniewska-Dziubinska, M.M.; Zielenkiewicz, P.; Siedlecki, P. Development and evaluation of a deep learning model for protein–ligand binding affinity prediction. *Bioinformatics* **2018**, *34*, 3666–3674. [CrossRef] [PubMed]
14. Wang, K.; Zhou, R.; Li, Y.; Li, M. DeepDTAF: A deep learning method to predict protein–ligand binding affinity. *Brief. Bioinform.* **2021**, *22*, bbab072. [CrossRef] [PubMed]
15. Ganapathiraju, M.K.; Klein-Seetharaman, J.; Balakrishnan, N.; Reddy, R. Characterization of protein secondary structure. *IEEE Signal Process. Mag.* **2004**, *21*, 78–87. [CrossRef]
16. Weininger, D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *J. Chem. Inf. Comput. Sci.* **1988**, *28*, 31–36. [CrossRef]
17. Wang, Z.; Zheng, L.; Liu, Y.; Qu, Y.; Li, Y.Q.; Zhao, M.; Mu, Y.; Li, W. OnionNet-2: A convolutional neural network model for predicting protein-ligand binding affinity based on residue-atom contacting shells. *Front. Chem.* **2021**, *9*, 753002. [CrossRef]
18. Wang, Y.; Wu, S.; Duan, Y.; Huang, Y. A point cloud-based deep learning strategy for protein–ligand binding affinity prediction. *Brief. Bioinform.* **2022**, *23*, bbab474. [CrossRef]
19. Öztürk, H.; Özgür, A.; Ozkirimli, E. DeepDTA: Deep drug–target binding affinity prediction. *Bioinformatics* **2018**, *34*, i821–i829. [CrossRef]

20. Rezaei, M.A.; Li, Y.; Wu, D.; Li, X.; Li, C. Deep Learning in Drug Design: Protein-Ligand Binding Affinity Prediction. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **2020**, *19*, 407–417. [CrossRef]
21. Abbasi, K.; Razzaghi, P.; Poso, A.; Amanlou, M.; Ghasemi, J.B.; Masoudi-Nejad, A. DeepCDA: Deep cross-domain compound-protein affinity prediction through LSTM and convolutional neural networks. *Bioinformatics* **2020**, *36*, 4633–4642. [CrossRef]
22. Nguyen, T.; Le, H.; Quinn, T.P.; Nguyen, T.; Le, T.D.; Venkatesh, S. GraphDTA: Predicting drug–target binding affinity with graph neural networks. *Bioinformatics* **2021**, *37*, 1140–1147. [CrossRef] [PubMed]
23. Yuan, W.; Chen, G.; Chen, C.Y.C. FusionDTA: Attention-based feature polymerizer and knowledge distillation for drug-target binding affinity prediction. *Brief. Bioinform.* **2022**, *23*, bbab506. [CrossRef] [PubMed]
24. Hua, Y.; Song, X.; Feng, Z.; Wu, X. MFR-DTA: A multi-functional and robust model for predicting drug-target binding affinity and region. *Bioinformatics* **2023**, *39*, btad056. [CrossRef] [PubMed]
25. Jin, Z.; Wu, T.; Chen, T.; Pan, D.; Wang, X.; Xie, J.; Quan, L.; Lyu, Q. CAPLA: Improved prediction of protein–ligand binding affinity by a deep learning approach based on a cross-attention mechanism. *Bioinformatics* **2023**, *39*, btad049. [CrossRef]
26. Zhu, Y.; Zhao, L.; Wen, N.; Wang, J.; Wang, C. DataDTA: A multi-feature and dual-interaction aggregation framework for drug–target binding affinity prediction. *Bioinformatics* **2023**, *39*, btad560. [CrossRef] [PubMed]
27. Kak, S.C. Quantum neural computing. *Adv. Imaging Electron Phys.* **1995**, *94*, 259–313.
28. Smith, G.D.; Steele, N.C.; Albrecht, R.F.; Ventura, D.; Martinez, T. An artificial neuron with quantum mechanical properties. In *Artificial Neural Nets and Genetic Algorithms: Proceedings of the International Conference in Norwich, UK, 1997*; Springer: Vienna, Austria, 1998; pp. 482–485.
29. Zhou, R.; Qin, L.; Jiang, N. Quantum perceptron network. In *Artificial Neural Networks–ICANN 2006: 16th International Conference, Athens, Greece, 10–14 September 2006*; Proceedings, Part I 16; Springer: Berlin/Heidelberg, Germany, 2006; pp. 651–657.
30. Lloyd, S.; Mohseni, M.; Rebentrost, P. Quantum algorithms for supervised and unsupervised machine learning. *arXiv* **2013**, arXiv:1307.0411.
31. Wiebe, N.; Kapoor, A.; Svore, K.M. Quantum deep learning. *arXiv* **2014**, arXiv:1412.3489. [CrossRef]
32. Matsui, N.; Takai, M.; Nishimura, H. A network model based on qubitlike neuron corresponding to quantum circuit. *Electron. Commun. Jpn. (Part III Fundam. Electron. Sci.)* **2000**, *83*, 67–73. [CrossRef]
33. Preskill, J. Quantum computing in the NISQ era and beyond. *Quantum* **2018**, *2*, 79. [CrossRef]
34. Cerezo, M.; Arrasmith, A.; Babbush, R.; Benjamin, S.C.; Endo, S.; Fujii, K.; McClean, R.J.; Mitarai, K.; Yuan, X.; Cincio, L.; et al. Variational quantum algorithms. *Nat. Rev. Phys.* **2021**, *3*, 625–644. [CrossRef]
35. Tacchino, F.; Macchiavello, C.; Gerace, D.; Bajoni, D. An artificial neuron implemented on an actual quantum processor. *NPJ Quantum Inf.* **2019**, *5*, 26. [CrossRef]
36. Zhao, J.; Zhang, Y.H.; Shao, C.P.; Wu, Y.C.; Guo, G.C.; Guo, G.P. Building quantum neural networks based on a swap test. *Phys. Rev. A* **2019**, *100*, 012334. [CrossRef]
37. Cong, I.; Choi, S.; Lukin, M.D. Quantum convolutional neural networks. *Nat. Phys.* **2019**, *15*, 1273–1278. [CrossRef]
38. Pesah, A.; Cerezo, M.; Wang, S.; Volkoff, T.; Sornborger, A.T.; Coles, P.J. Absence of barren plateaus in quantum convolutional neural networks. *Phys. Rev. X* **2021**, *11*, 041011. [CrossRef]
39. Zhao, C.; Gao, X.S. QDNN: Deep neural networks with quantum layers. *Quantum Mach. Intell.* **2021**, *3*, 15. [CrossRef]
40. Huang, H.Y.; Kueng, R.; Preskill, J. Information-Theoretic Bounds on Quantum Advantage in Machine Learning. *Phys. Rev. Lett.* **2021**, *126*, 190505. [CrossRef]
41. Huang, H.Y.; Broughton, M.; Cotler, J.; Chen, S.; Li, J.; Mohseni, M.; Neven, H.; Babbush, R.; Kueng, R.; Preskill, J.; et al. Quantum advantage in learning from experiments. *Science* **2022**, *376*, 1182–1186. [CrossRef] [PubMed]
42. Abbas, A.; Sutter, D.; Zoufal, C.; Lucchi, A.; Figalli, A.; Woerner, S. The power of quantum neural networks. *Nat. Comput. Sci.* **2021**, *1*, 403–409. [CrossRef]
43. Xiao, T.; Zhai, X.; Wu, X.; Fan, J.; Zeng, G. Practical advantage of quantum machine learning in ghost imaging. *Commun. Phys.* **2023**, *6*, 171. [CrossRef]
44. Qu, Z.; Li, Y.; Tiwari, P. QNMF: A quantum neural network based multimodal fusion system for intelligent diagnosis. *Inf. Fusion* **2023**, *100*, 101913. [CrossRef]
45. Forestano, R.T.; Comajoan Cara, M.; Dahale, G.R.; Dong, Z.; Gleyzer, S.; Justice, D.; Kong, K.; Magorsch, T.; Matchev, T.K.; Matcheva, K.; et al. A Comparison between Invariant and Equivariant Classical and Quantum Graph Neural Networks. *Axioms* **2024**, *13*, 160. [CrossRef]
46. Li, J.; Alam, M.; Congzhou, M.S.; Wang, J.; Dokholyan, N.V.; Ghosh, S. Drug discovery approaches using quantum machine learning. In Proceedings of the 2021 58th ACM/IEEE Design Automation Conference, San Francisco, CA, USA, 5–9 December 2021; pp. 1356–1359.
47. Li, J.; Topaloglu, R.O.; Ghosh, S. Quantum generative models for small molecule drug discovery. *IEEE Trans. Quantum Eng.* **2021**, *2*, 1–8. [CrossRef]
48. Sagingalieva, A.; Kordzanganeh, M.; Kenbayev, N.; Kosichkina, D.; Tomashuk, T.; Melnikov, A. Hybrid quantum neural network for drug response prediction. *Cancers* **2023**, *15*, 2705. [CrossRef]
49. Domingo, L.; Djukic, M.; Johnson, C.; Borondo, F. Binding affinity predictions with hybrid quantum-classical convolutional neural networks. *Sci. Rep.* **2023**, *13*, 17951. [CrossRef]

50. Avramouli, M.; Savvas, I.K.; Vasilaki, A.; Garani, G. Unlocking the potential of quantum machine learning to advance drug discovery. *Electronics* **2023**, *12*, 2402. [CrossRef]
51. 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*, 302–309. [CrossRef]
52. Burley, S.K.; Berman, H.M.; Bhikadiya, C.; Bi, C.; Chen, L.; Di Costanzo, L.; Zardecki, C. RCSB Protein Data Bank: Biological macromolecular structures enabling research and education in fundamental biology, biomedicine, biotechnology and energy. *Nucleic Acids Res.* **2019**, *47*, D464–D474. [CrossRef]
53. McClean, J.R.; Boixo, S.; Smelyanskiy, V.N.; Babbush, R.; Neven, H. Barren plateaus in quantum neural network training landscapes. *Nat. Commun.* **2018**, *9*, 4812. [CrossRef] [PubMed]
54. Pérez-Salinas, A.; Cervera-Lierta, A.; Gil-Fuster, E.; Latorre, J.I. Data re-uploading for a universal quantum classifier. *Quantum* **2020**, *4*, 226. [CrossRef]
55. Schuld, M.; Sweke, R.; Meyer, J.J. Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Phys. Rev. A* **2021**, *103*, 032430. [CrossRef]
56. LaRose, R.; Coyle, B. Robust data encodings for quantum classifiers. *Phys. Rev. A* **2020**, *102*, 032420. [CrossRef]
57. Chesher, D. Evaluating assay precision. *Clin. Biochem. Rev.* **2008**, *29* (Suppl. S1), S23.
58. Ketkar, N.; Moolayil, J.; Ketkar, N.; Moolayil, J. *Deep Learning with Python: Learn Best Practices of Deep Learning Models with PyTorch*, 2nd ed.; Apress: Berkeley, CA, USA, 2020.
59. Bergholm, V.; Izaac, J.; Schuld, M.; Gogolin, C.; Ahmed, S.; Ajith, V.; Killoran, N. Pennylane: Automatic differentiation of hybrid quantum-classical computations. *arXiv* **2018**, arXiv:1811.04968.
60. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Duchesnay, É. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
61. Loshchilov, I.; Hutter, F. Decoupled weight decay regularization. *arXiv* **2017**, arXiv:1711.05101.